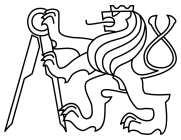
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Vizualizace doménově specifického jazyka pro popis zpracování časových řad
Student:	Bc. Jakub Klíma
Vedoucí:	Ing. Ondřej Malík
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2017/18

Pokyny pro vypracování

Pro stávající aplikaci byl vyvinut doménově specifický jazyk, který slouží pro popis zpracování časových řad uložených v datových strukturách. Jazyk umožňuje časové řady třídit, agregovat a provádět s nimi další operace.

V případě rozsáhlých datových struktur nebo potřeby provést velké množství operací je popis v jazyce složitý a jeho ruční tvorba je obtížná a náchylná k chybám. Cílem práce je vytvořit GUI, které tento jazyk vizualizuje, čímž usnadní tvorbu těchto popisů vývojářům a zároveň ji zpřístupní i běžným uživatelům.

Pokyny pro vypracování:
- Nastudujte možné způsoby vizualizace podobných jazyků.
- Naimplementujte GUI pro vizualizaci tohoto jazyka umožňující:
  - definovat datové pohledy pro výběr, třídění a agregace časových řad,
  - skládání datových pohledů do stromových struktur,
  - mapování datových pohledů do tabulek a grafů k zobrazení.
- Dbejte na čitelnost a znovupoužitelnost kódu.
- Veškerá veřejná rozhraní tříd řádně zdokumentujte.

Seznam odborné literatury

Jakub Klíma, *Implementace modulu pro zpracování datových struktur*, Bakalářská práce, ČVUT, 2015
Robert C. Martin, *Clean Code: A Handbook of Agile Software Craftsmanship.* Prentice Hall, 2009
Steve McConnell, *Code Complete, Second Edition.* Microsoft Press, Redmond, WA, USA, 2004

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 23. prosince 2016

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Diplomová práce

Vizualizace doménově specifického jazyka pro popis zpracování časových řad

Bc. Jakub Klíma

Vedoucí práce: Ing. Ondřej Malík, Ph.D.

3. května 2017

Poděkování

Děkuji svému vedoucímu práce Ing. Ondřeji Malíkovi, Ph.D. za přínosné konzultace a rady k plánování práce, k návrhovým rozhodnutím při implementaci a k obsahu a formě textu práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. Dále prohlašuji, že jsem s Českým vysokým učením technickým v Praze uzavřel dohodu, na základě níž se ČVUT vzdalo práva na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona. Tato skutečnost nemá vliv na ust. § 47b zákona č. 111/1998 Sb., o vysokých školách, ve znění pozdějších předpisů.

V Praze dne 3. května 2017 .....

Odkaz na tuto práci

Klíma, Jakub. *Vizualizace doménově specifického jazyka pro popis zpracování časových řad.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Tato práce se zabývá tvorbou grafického uživatelského rozhraní (GUI) pro vizualizaci doménově specifického jazyka vyvinutého pro popis zpracování časových řad v datových strukturách v rámci stávající aplikace. Jazyk umožňuje časové řady třídit, agregovat a provádět s nimi další operace. V případě rozsáhlých datových struktur nebo potřeby provést velké množství operací je popis v jazyce složitý a jeho ruční tvorba je obtížná a náchylná k chybám. GUI tvorbu těchto popisů usnadní a zároveň ji zpřístupní i běžným uživatelům. Tvorbě GUI předchází rešerše existujících nástrojů, které řeší podobnou problematiku. V práci byl kladen důraz na kvalitu implementace, především proto, že je nástroj součástí aplikace, na které pracují i další vývojáři. Implementace nástroje byla úspěšně dokončena a součástí práce je i její dokumentace. V současné době je tento nástroj používán ostatními vývojáři aplikace a do budoucna se pak plánuje jeho zpřístupnění běžným uživatelům aplikace a jeho další rozšiřování a vylepšování.

Klíčová slova datové pohledy, DSL, grafy, GUI, implementace, stromové struktury, tabulky, XML, zpracování dat

Abstract

This thesis deals with the development of a graphical user interface (GUI) for visualization of a domain-specific language developed to describe time series processing in data structures within an existing application. The language allows to sort, aggregate, and perform other operations with time series. In the case of large data structures or when large number of operations with the data structure need to be performed, the description in the language is complex and its manual creation is difficult and error prone. The GUI makes creating of these descriptions easier and also makes it accessible to common users. The GUI implementation is preceded by a survey of existing tools that address similar problem. Emphasis was placed on quality of the implementation, especially since the tool is a part of the application that other developers are working on. The implementation of the tool has been successfully completed and its documentation is part of the thesis. At this time, the tool is used by other application developers, and in future, it is expected that the tool will be made available to common users and it will be further enhanced.

Keywords data views, DSL, charts, graphs, GUI, implementation, tree structures, tables, XML, data processing

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

Tato práce se zabývá tvorbou grafického uživatelského rozhraní (GUI) pro vizualizaci doménově specifického jazyka. Jazyk byl vyvinut pro popis zpracování časových řad v datových strukturách v rámci stávající aplikace. Tato aplikace se zabývá výpočty a modelováním v oblasti energetiky a je již delší dobu používána a zároveň průběžně rozšiřována.

Čtenář práce je nejprve seznámen s jazykem pro zpracování časových řad a dalšími podrobnostmi nutnými k porozumění práce, dále je pak provedena rešerše některých existujících nástrojů, zabývajících se podobnou problematikou a nakonec následuje popis návrhu, implementace a dokumentace vytvořeného nástroje.

Úvodní kapitola popisuje problematiku zpracování a zobrazení dat ve stávající aplikaci a dále motivaci k vypracování této práce, tedy k implementaci GUI. Následuje popis konceptů nutných k základnímu porozumění této problematiky.

1.1 Data a datové objekty

Data, která budu popisovat, jsou povahy **časových řad** – jedná se o posloupnost číselných hodnot s časovou osou. Platí, že každá hodnota má svůj časový interval platnosti a tyto intervaly platnosti se nepřekrývají. Pro zjednodušení si lze představit například časovou řadu v délce jednoho dne se 24 hodnotami ke každé hodině.

Tyto časové řady jsou uloženy v **datových objektech**. Datové objekty kromě časových řad mohou obsahovat i další datové objekty. Jako příklad datového objektu si lze představit e-shop Alza, který bude mít dvě časové řady – počet návštěv a počet objednávek v danou hodinu dne 1. 1. 2017 (viz tabulka 1.1).

Časové řady mají dále parametry, které umožňují je identifikovat. Jeden z těchto parametrů je například název nebo identifikátor datového objektu, ke

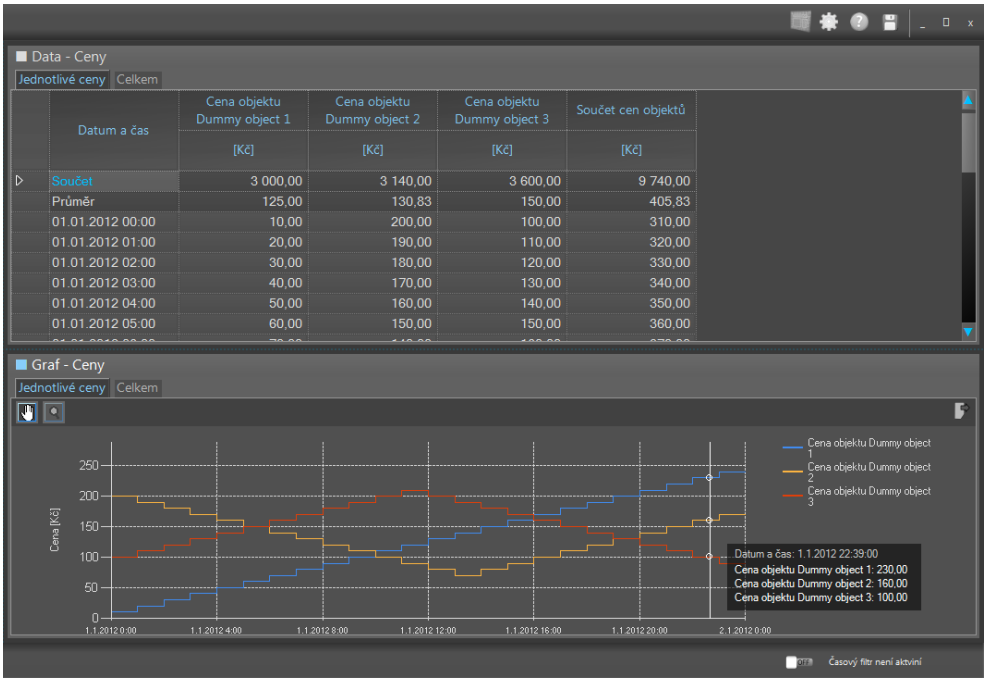
Tabulka 1.1: Ukázka datového objektu e-shop Alza

Hodina	Návštěvníků	Objednávek
0:00	2000	134
1:00	1000	67
2:00	500	34
3:00	500	34
4:00	500	34
5:00	2000	134
6:00	3000	200
7:00	5000	334
8:00	10000	667
9:00	11000	734
10:00	10000	667
11:00	9000	600
12:00	10000	667
13:00	11000	734
14:00	10000	667
15:00	9000	600
16:00	10000	667
17:00	11000	734
18:00	10000	667
19:00	9000	600
20:00	7000	467
21:00	6000	400
22:00	5000	334
23:00	4000	267

kterému patří.

Každý datový objekt má svůj typ, který bude označován jako **datová třída**. Z hlediska implementace se skutečně jedná o třídu v programovacím jazyce a datovým objektem pak mám na mysli instanci této třídy. V případě e-shopu Alza by mohla být datová třída E-shop a další objekty (instance) této třídy by mohly být např. e-shop Czc a e-shop Mironet.

Všechny časové řady v daném datovém objektu i všech jeho vnořených objektů (rekurzivně) se vztahující ke stejnému časovému období – resp. sdílí **společnou časovou osu**. V případě e-shopu Alza, to znamená, že obě časové řady obsahují hodinová data a to pro stejné období 1. 1. 2017.



Obrázek 1.1: Zobrazovací komponenty v naší aplikaci

1.2 Potřeba zobrazit data v přehledné formě

Datové objekty obsahují typicky rozsáhlá data – desítky až stovky časových řad, které mají až 500 tisíc hodnot (v případě minutových ročních dat). Zobrazit je najednou v zobrazovacích komponentách aplikace by bylo velmi nepřehledné.

Platí, že najednou je zobrazen vždy buď jeden datový objekt, nebo skupina datových objektů stejného typu, kde navíc všechny objekty ve skupině mají stejnou časovou osu. Zobrazená data tedy budou mít vždy jednu **společnou časovou osu**, což má důležité implikace pro zobrazovací komponenty.

Mezi zobrazovací komponenty (viz obr. 1.1) patří:

1. **Datagridy** – komponenty zobrazující tabulky dat. Díky společné časové ose časových řad, jsou zobrazeny hodnoty jednotlivých řad ve sloupcích a společná časová osa tvoří indexační (první) sloupec.

2. **Spojnicové grafy** – časové řady jsou v grafu zobrazeny jako jednotlivé průběhy a společná časová osa definuje v grafu osu X.

Grafy i datagridy jsou zobrazeny ve vlastním **panelu**. Pokud je grafů nebo datagridů zobrazeno najednou více, lze mezi nimi v daném panelu přepínat pomocí **záložek** (viz obr. 1.1).

1.3 Modul pro zpracování a zobrazení dat

Řešením potřeby zobrazení dat v přehledné formě je modul pro zpracování a zobrazení dat (dále **Modul**), který se zabývá následujícím:

1. Načítá data z datových objektů.

2. Připravuje data do vhodné formy, pomocí tzv. **datových pohledů**. Tyto datové pohledy umožňují provádět nad daty operace jako třídění, agregování, řazení a dále i např. zamčení k editaci. Podrobnější vysvětlení, co to jsou datové pohledy, naleznete v kapitole 3.

3. Připravuje tabulky a spojnicové grafy na základě těchto datových pohledů.

Tento Modul vznikl jako má bakalářská práce [1]. Diplomová práce je rozšířením této bakalářské práce a uvádí z ní jen takové detaily, které jsou potřebné k jejímu porozumění. Ostatní detaily je v případě zájmu možno najít v bakalářské práci.

1.4 Předpisy a interní jazyk

V aplikaci je možné definovat **předpis pro přípravu dat dané datové třídy**, který Modul umí zpracovat (dále **Předpis**). Předpisy jsou psány v interně vyvinutém doménově specifickém jazyce (dále **Jazyk**), který je odvozen z jazyka XML – Předpisy jsou pak jednotlivé XML dokumenty. Jazyk umožňuje:

1. identifikovat časové řady datového objektu,

2. vytvářet datové pohledy a vzájemně je propojovat (bude vysvětleno v kapitole 3),

3. určit co bude zobrazeno uživateli ve výsledných tabulkách a grafech, tedy určit jejich zdrojové datové pohledy a další parametry.

Předpis může být buď pro jeden datový objekt, nebo pro skupinu datových objektů stejné datové třídy (např. skupinu e-shopů Alza, Czc a Mironet). To je potřeba rozlišit, protože často je nutné, zobrazit data v jiné formě při náhledu na jeden objekt než při náhledu na skupinu objektů. Skupiny mají zpravidla složitější Předpisy, ve kterých se hojně využívá třídění, seskupování a agregací dat.

```
<tree treeId="podstromNakladyTrhuSpojene">
  <orderBy viewId="nakladyTrhuSpojene" orderingCriteria="classInstanceId">
    <merge viewId="nakladyTrhuSpojeneNeserazeneDleInstance">
      <subtree treeId="podstromNakladyNaDriveKoupenou..." />
      <groupByAggregation viewId="nakladyTrhuCelkem"
                          aggregationId="nakladyTrhuCelkem"
                          operation="addition"
                          groupingParameter="classInstanceId"
        columnNameBlueprintResource="Economics_NakladyTrhuCelkem">
        <subtree treeId="podstromNakladyNaDriveKoupenou..." />
      </groupByAggregation>
    </merge>
  </orderBy>
</tree>
```

Obrázek 1.2: Ukázka složitosti interního předpisu

1.5 Problémy s ručně psanými předpisy

V současné době vývojáři aplikace tvoří Předpisy manuálně. Obzvláště v případě rozsáhlých datových tříd nebo potřeby provést velké množství operací jsou Předpisy **složité** a jejich ruční tvorba je obtížná a náchylná k chybám.

Dále je do budoucna požadavek umožnit tvorbu Předpisů i uživatelům aplikace, což při současné složitosti Předpisů není možné právě díky složitosti předpisů.

Pro představu složitosti Předpisů uvádím ukázku z jednoho z dosud nejsložitějších, nicméně používaných Předpisů (viz obr. 1.2). Jedná se o Předpis pro datový objekt obsahující souhrnné zobrazení výpočetního modelu.

Celý Předpis má zhruba 700 řádek, přičemž přibližně:

- 200 řádek popisuje identifikaci časových řad datového objektu,
- 400 řádek popisuje tvorbu datových pohledů a jejich propojení,
- 100 řádek popisuje datagridy a grafy.

Zobrazený úsek Předpisu ukazuje část z tvorby datových pohledů – je to jeden ze zhruba 40 stromů datových pohledů, které popisují jejich tvorbu a vzájemné napojení.

Je tedy zřejmé, že manuální tvorba těchto Předpisů je obtížná a těžko udržitelná.

Kapitola 2

Cíl práce

Z úvodní kapitoly jsou zřejmé nedostatky ruční tvorby Předpisů pro zpracování a zobrazení dat v popsaném interně vyvinutém Jazyce.

Cílem práce je tedy vytvořit grafické uživatelské rozhraní (GUI), které Jazyk vizualizuje, čímž usnadní tvorbu Předpisů vývojářům a zároveň ji zpřístupní i běžným uživatelům.

Postup vypracování bude následující:

1. Nastudování možných způsobů vizualizace DSL podobných Jazyku (rešeršní část práce).
2. Návrh a implementace GUI pro vizualizaci Jazyka umožňující:
   - definovat datové pohledy pro výběr, třídění a agregace časových řad,
   - skládání datových pohledu do stromových struktur,
   - mapování datových pohledu do tabulek a grafů k zobrazení.
3. Dokumentace veškerého veřejného rozhraní tříd.

Při implementaci GUI dále bude kladen důraz na čitelnost a znovupoužitelnost kódu, neboť je součástí aplikace, na které pracují i další programátoři a je tedy pravděpodobné, že bude dále rozšiřován či drobně opravován a upravován a to nejen mnou, ale i ostatními programátory.

Kapitola 3

Popis interního jazyka a předpisů

Interní Jazyk a Předpisy byly již stručně představeny, tato kapitola se však věnuje jejich podrobnějšímu popisu a vysvětlení, které je potřebné k plnému porozumění této práci.

Tato kapitola čerpá z mojí bakalářské práce [1], která se tvorbou Předpisů a Jazyka zabývala.

3.1 Nové koncepty

Před podrobnějším popisem Jazyka a struktury Předpisu, je nutné zadefinovat nové koncepty / pojmy. V této podkapitole naleznete popis těchto konceptů na úrovni potřebné k pochopení Jazyka a Předpisů. Podrobnější popis těchto konceptů a jejich interního chování lze nalézt v mé bakalářské práci [1].

3.1.1 Datové pohledy

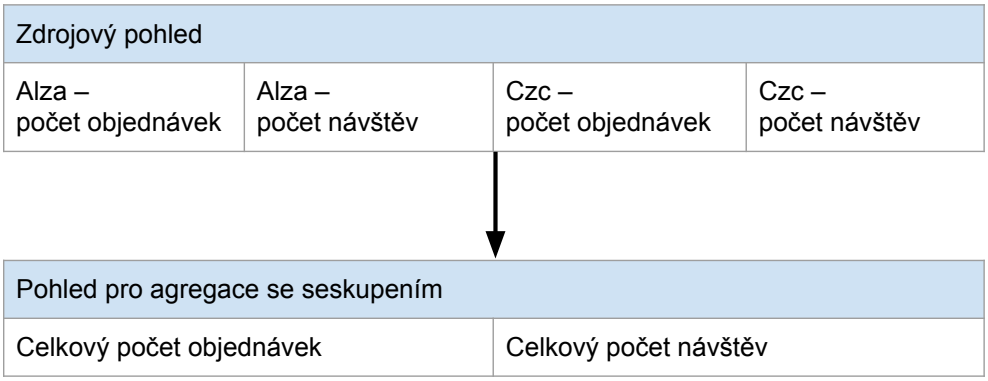
Datový pohled si lze představit jako černou krabičku, která provádí danou operaci (agregaci, řazení apod. viz dále) nad časovými řadami. Na vstupu krabičky je jiný datový pohled, resp. jeho výstupní časové řady. Na výstupu krabičky jsou časové řady po provedení operace.

Vstupní datový pohled je vždy jen jeden, až na výjimku u pohledu pro sloučení pohledů, který může mít na vstupu datových pohledů více (viz dále).

Následuje stručný popis jednotlivých datových pohledů:

1. **Pohled pro výběr** – vybere ze zdrojového pohledu podmnožinu časových řad.

2. **Pohled pro agregace** – provede nad vstupními časovými řadami aritmetickou operaci a výstupem je jedna výsledná časová řada.



Obrázek 3.1: Pohled pro agregace se seskupením

3. **Pohled pro sloučení pohledů** – slučuje časové řady z několika pohledů do jednoho. Na vstupu tedy nemá právě jeden datový pohled jako ostatní pohledy (a právě díky tomu není struktura pohledů striktně stromová).

4. **Pohled pro agregace se seskupením** – podobně jako pohled pro agregace provede nad vstupními časovými řadami aritmetickou operaci, nicméně složitějším způsobem. Vstupní časové řady jsou nejdříve rozřazeny do skupin a aritmetická operace je aplikována až na tyto jednotlivé skupiny. Výsledná časová řada je potom jedna pro každou skupinu. Pohledu je nutné předepsat parametr, podle kterého dojde k rozdělení do skupin – např. rozdělení do skupin podle instance datového objektu, ze kterého časová řada pochází. Příklad tohoto pohledu je vidět na obr. 3.1.

5. **Pohled pro řazení** – umožňuje řadit vstupní časové řady dle zadaného parametru – např. seřadit časové řady podle instance datového objektu, ze kterého pochází.

6. **Pohled pro otočení znaménka** – otočí znaménko všem číselným hodnotám vybraných časových řad. Časové řady jsou vybrány opět pomocí selekcí časových řad.

7. **Pohled pro přejmenování** – přejmenuje vstupní časové řady.

8. **Pohled pro zamknutí** – zamkne vybrané časové řady, aby je nebylo možné v datagridu editovat. Časové řady jsou i v tomto případě vybrány pomocí selekcí časových řad.

3.1.2 Pojmenování datových pohledů

Pro datové pohledy existuje český název, název třídy v programovacím jazyce a název XML elementu, které reprezentuje daný datový pohled v Předpisu.

Tabulka 3.1: Názvy datových pohledů

Celý název	Název třídy	Název XML elementu
Pohled pro výběr	SelectionView	selection
Pohled pro agregace	AggregationView	aggregation
Pohled pro sloučení pohledů	MergeView	merge
Pohled pro agregace se seskupením	GroupByAggregationView	groupByAggregation
Pohled pro řazení	OrderByView	orderBy
Pohled pro otočení znaménka	ReverseSignView	reverseSign
Pohled pro přejmenování	RenameView	rename
Pohled pro zamknutí	ReadOnlyView	readOnly

Třídy v programovacím jazyce mají vždy příponu **View**. Název XML elementu je pak identický jako název třídy, jen chybí zmíněná přípona kvůli redundanci, neboť v Předpise jsou datové pohledy obaleny XML elementem views (viz dále).

Tabulka 3.1 zobrazuje všechny tři tyto názvy zmíněných datových pohledů, což je užitečné pro jejich použití ve zbytku práce.

3.1.3 Zdrojová tabulka

Zdrojová tabulka je speciální datový pohled. Vstupem této tabulky není datový pohled jako u ostatních datových pohledů, ale vstupní datové objekty (případně jeden). Výstupem jsou pak všechny časové řady těchto datových objektů.

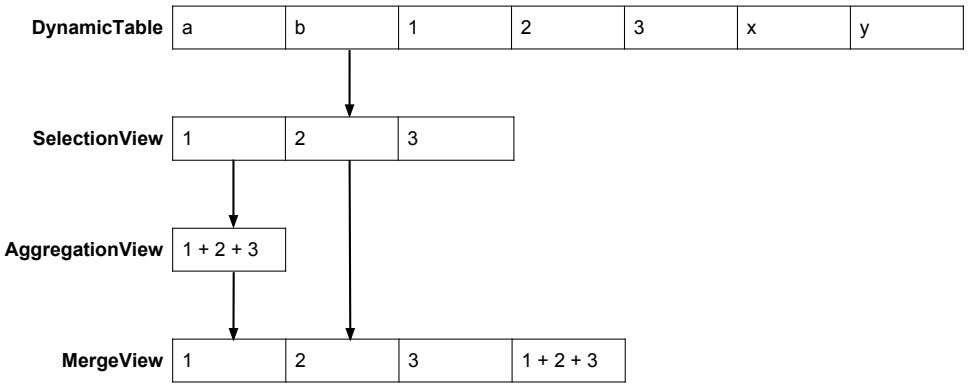
Tabulka funguje jako zdroj dat pro všechny ostatní pohledy a převádí tak vstupní datové objekty na datový pohled.

Pro úplnost k tabulce 3.1 je název třídy zdrojové tabulky – **DynamicTable** a název odpovídajícího XML elementu – **table**.

3.1.4 Propojení datových pohledů

Propojení datových pohledů vede přirozeně na stromovou strukturu – za předpokladu, že datový pohled má jeden vstupní datový pohled. Kvůli pohledu pro sloučení dat a jeho více vstupním pohledům tedy ne přímo strom, ale pouze orientovaný acyklický graf (viz obr. 3.2). Pro zjednodušení ale bude struktura označována jako stromová.

Kořeny těchto stromových struktur jsou pak vždy zdrojové tabulky. Listy těchto struktur jsou typicky cílové datové pohledy, které jsou zvoleny k zobrazení v grafech a datagridech.



Obrázek 3.2: Propojení datových pohledů

3.1.5 Příklad

Pro lepší představu je zde uveden příklad stromu datových pohledů (viz obr. 3.2). Cílem tohoto stromu datových pohledů je vybrat určité časové a zobrazit k nim jejich součet.

Ve zdrojové tabulce jsou i časové řady, které nechceme zobrazit, proto pomocí pohledu pro výběr vybereme pouze potřebné časové řady. Dále chceme zobrazit součet vybraných časových řad, připravíme si jej tedy pomocí pohledu pro agregace. Ve výsledném zobrazení chceme mít jak vybrané časové řady, tak řadu s jejich součtem, sloučíme tedy datové pohled pro výběr a pohled pro agregace.

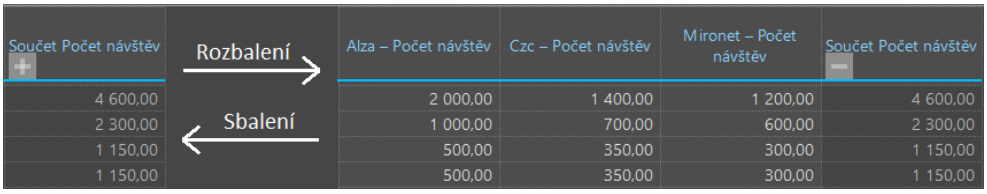
Tím dostáváme cílový pohled, který lze například zobrazit v datagridu.

3.1.6 Selekce časových řad

Jak již bylo řečeno, časové řady obsahují parametry, kterými je lze identifikovat. Tyto parametry mohou být např. datová třída, ve které je časová řada obsažena nebo vlastnost datové třídy, ve které se časová řada nachází (resp. cesta k této vlastnosti v hierarchii datových objektů). Selekce časových řad tyto parametry využívají, obzvláště pak tzv. běžná selekce (viz dále).

Selekci časových řad si lze představit jako černou krabičku, která má na vstupu množinu časových řad a na výstupu vybranou podmnožinu těchto časových řad, které prošly sítem. Síto je řízeno výběrovým mechanismem, resp. zobrazením: časová řada $\rightarrow$ ANO/NE, určující, zda časová řada sítem prošla.

Existuje několik druhů selekcí a každá má jiný výběrový mechanismus. Pro tuto práci je důležitá hlavně tzv. běžná selekce. Běžná selekce obsahuje sadu zmíněných parametrů a očekávaných hodnot, a aby časová řada byla přijata, musí všechny hodnoty jejich parametrů odpovídat očekávaným hodnotám selekce. Rozbor dalších selekcí lze v případě zájmu nalézt v bakalářské práci [1].



Obrázek 3.3: Rozbalovací skupiny sloupců v datagridu

3.1.7 Seskupení sloupců v datagridu

V datagridu lze sloupce organizovat do skupin, které je možné rozbalovat a sbalovat (viz obr. 3.3). Tyto skupiny tak umožňují přehlednější zobrazení více sloupců v jednom datagridu.

Skupiny mohou být libovolně vnořeny a tvoří tedy stromové struktury (teď již bez výjimek, na rozdíl od stromové struktury datových pohledů).

3.2 Struktura předpisu, ukázka

Tato a následující podkapitola 3.3 je převzata z mé bakalářské práce [1] a je zde uvedena v původním znění pouze s drobným přizpůsobením této práci.

Podkapitola se zabývá rozborem struktury Předpisu. Nejprve bude předveden ukázkový XML soubor pro lepší ilustraci – konkrétně jeho vybrané části. Poté následuje podrobnější rozbor struktury jednotlivých sekcí Předpisu.

Na následující ukázce (viz obr. 3.4) je nastíněn význam a struktura nejdůležitějších částí Předpisu.

Elementy nejvyšší úrovně rozdělují Předpis na jednotlivé části. Elementy **chartStyles**, **datagridStyles** a **datagridColumnStyles** předepisují parametry zobrazovacích komponent. Jejich obsah je v této ukázce pro zjednodušení skryt. Ostatní části Předpisu pak souvisí hlavně s datovými pohledy. Element **tables** předepisuje zdrojové tabulky a element **columnSelections** předepisuje selekce časových řad. V této ukázce je jejich obsah také skryt.

Zadání struktury datových pohledů a jejich navázání na jednotlivé zobrazovací komponenty k tomuto Předpisu je znázorněno v následujícím diagramu (viz obr. 3.5).

Kořenem struktury je zdrojová tabulka a na ní jsou navázány dva pohledy pro výběr časových řad.

Zdrojová tabulka je definována v elementu **tables**. V elementu **trees** je pak část popisující strukturu datových pohledů. V této části se nachází definice pohledů pro výběr časových řad, použité selekce časových řad a jejich navázání na zdrojovou tabulku (bude podrobněji vysvětleno v následující podkapitole).

Předpis jednotlivých grafů, datagridů a jejich záložek je v elementech **chartViews** a **datagridViews**. Každá záložka zobrazovací komponenty má odkaz na datový pohled a na parametry pro tuto komponentu.

```
<scheme>

  <tables>…</tables>

  <trees>

    <tree treeId="stromProGraf1">
      <selection viewId="pohledProGraf1" columnSelectionIds="sloupecA,sloupecE,sloupecD">
        <table tableId="zakladniTabulka" />
      </selection>
    </tree>

    <tree treeId="stromProGraf2">
      <selection viewId="pohledProGraf2" columnSelectionIds="sloupecB,sloupecC">
        <table tableId="zakladniTabulka" />
      </selection>
    </tree>

  </trees>

  <chartViews>
    <view viewId="pohledProGraf1" chartStyleId="stylGrafu" chartTabLabelResource="Záložka grafu 1" />
    <view viewId="pohledProGraf2" chartStyleId="stylGrafu" chartTabLabelResource="Záložka grafu 2" />
  </chartViews>

  <datagridViews>
    <view viewId="zakladniTabulka" datagridStyleId="stylDatagridu"
          datagridTabLabelResource="Záložka datagridu" />
  </datagridViews>

  <columnSelections>…</columnSelections>
  <chartStyles>…</chartStyles>
  <datagridStyles>…</datagridStyles>
  <datagridColumnStyles>…</datagridColumnStyles>

</scheme>
```

Obrázek 3.4: Ukázkový XML předpis

Jak pohledy, tak některé parametry jsou označeny identifikátory, aby se na ně bylo možné v jiných částech Předpisu odkazovat.

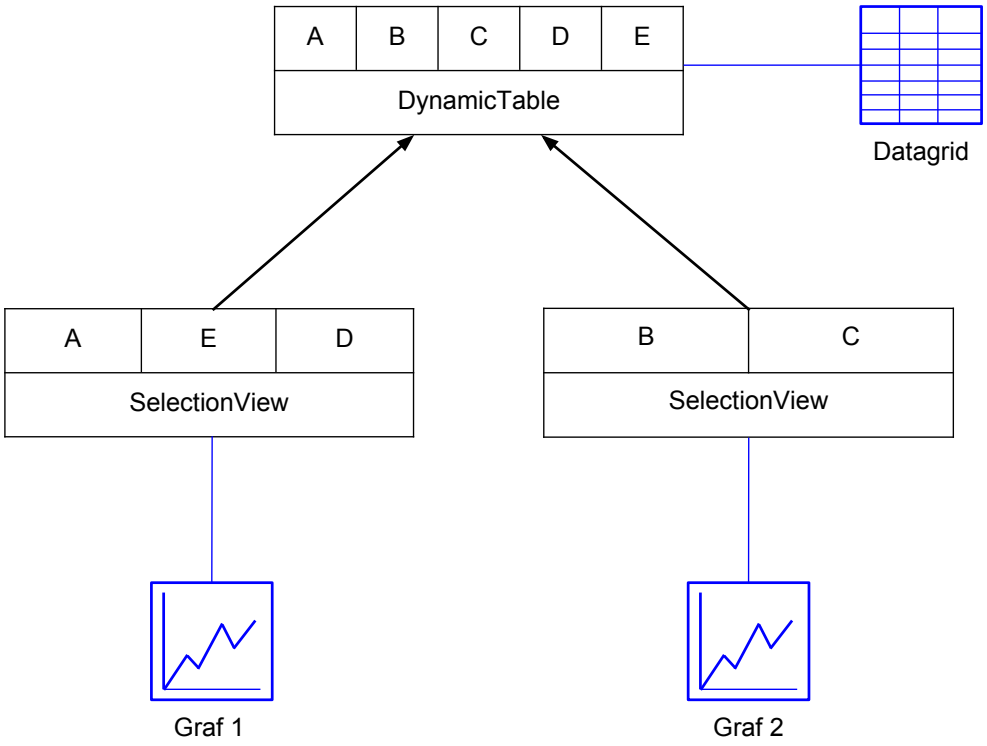
Úplný Předpis k této ukázce se nachází na přiloženém DVD.

3.3 Rozbor jednotlivých částí předpisu

Tato podkapitola se zabývá podrobnějším popisem jednotlivých částí Předpisu.

3.3.1 Selekce časových řad

V elementu **columnSelections** se nachází definice selekcí časových řad. Každá selekce má svůj identifikátor **columnSelectionId**, který umožní se na selekce odkazovat při definici datových pohledů. Selekce mají dále uveden svůj typ (viz podkapitola 3.1.6) a obsahují posloupnost identifikačních parametrů – jejich názvů a hodnot.



Obrázek 3.5: Strom pohledů s navěšením zobrazovacích komponent

3.3.2 Zdrojové tabulky dat

V předpisu je definována sada zdrojových tabulek, nad kterými lze stavět datové pohledy.

V této části Předpisu – element **tables**, se seznam těchto zdrojových tabulek vypíše a tabulkám se přidělí identifikátor **viewId**, který je následně využíván, aby bylo možné se na tabulky odkazovat při definici datových pohledů.

3.3.3 Stromy datových pohledů

V elementu **trees** se nachází předpis datových pohledů – jejich stromů (viz obr. 3.6).

Jednotlivé stromy jsou strukturovány tak, že jejich listy jsou zdrojové tabulky nebo jiné stromy (resp. kořeny těchto stromů), které se v každém patře výše obalují datovými pohledy.

Stromy mají svůj identifikátor **treeId**, díky kterému mohou být použity v definici jiných stromů jako jejich součást, resp. podstrom.

Jednotlivé datové pohledy jsou reprezentovány vlastním elementem. V definici pohledů lze používat zavedené selekce časových řad pomocí jejich identifi-

```
<trees>

  <tree treeId="stromProGraf1">
    <selection viewId="pohledProGraf1" columnSelectionIds="sloupecA,sloupecE,sloupecD">
      <table tableId="zakladniTabulka" />
    </selection>
  </tree>

  <tree treeId="stromProGraf2">
    <selection viewId="pohledProGraf2" columnSelectionIds="sloupecB,sloupecC">
      <table tableId="zakladniTabulka" />
    </selection>
  </tree>

</trees>
```

Obrázek 3.6: Sekce Trees v ukázkovém XML předpisu

kátorů **columnSelectionId** (viz posloupnost těchto identifikátorů v atributu *columnSelectionIds* v ukázce Předpisu 3.6). Každý datový pohled je označen identifikátorem **viewId**, díky kterému se na něj lze při definici grafů a datagridů odkazovat.

Kromě těchto parametrů, pak mohou mít datové pohledy další parametry, které jsou dány konkrétním typem pohledu.

3.3.4 Parametry pro graf

Parametry grafů jsou definovány v elementu **chartStyles**. Jednotlivé parametry jsou označeny identifikátorem **chartStyleId**, dle kterého se na ně při definici grafů odkazuje.

Grafům lze parametrizovat popisek osy Y a formátováci řetězec dat na ose X.

3.3.5 Styly sloupců datagridu

Styly jednotlivých sloupců pro datagridy jsou v elementu **datagridColumnStyles**. Každý styl má svůj identifikátor **datagridColumnStyleId**, podle kterého se na něj lze z definice parametrů datagridu odkazovat.

Sloupcům lze předepsat šířku, formátovací řetězec, zarovnání a zvýraznění.

3.3.6 Parametry datagridu

V elementu **datagridStyles** jsou definovány parametry datagridu. Každé parametry mají svůj identifikátor **datagridStyleId**, na základě kterého se lze na parametry odkázat z definice datagridů.

Parametry datagridu mají definované souřadnice ukotvené buňky, seznam výsledných sloupců s jejich styly a seskupení sloupců (viz podkapitola 3.1.7).

Seskupení sloupců má stromovou strukturu a je tedy přímo reprezentováno stromovou strukturou elementů.

```
<chartViews>
  <view viewId="pohledProGraf1" chartStyleId="stylGrafu" chartTabLabelResource="Záložka grafu 1" />
  <view viewId="pohledProGraf2" chartStyleId="stylGrafu" chartTabLabelResource="Záložka grafu 2" />
</chartViews>
```

Obrázek 3.7: Sekce ChartViews v ukázkovém XML předpisu

```
<datagridViews>
  <view viewId="zakladniTabulka" datagridStyleId="stylDatagridu"
        datagridTabLabelResource="Záložka datagridu" />
</datagridViews>
```

Obrázek 3.8: Sekce DatagridViews v ukázkovém XML předpisu

3.3.7 Jednotlivé grafy

V elementu **chartViews** jsou definovány jednotlivé grafy, respektive jejich záložky (viz obr. 3.7). Každý graf má odkaz na svůj zdrojový pohled – **viewId**, odkaz na svoje parametry – **chartStyleId** a dále popisek záložky.

3.3.8 Jednotlivé datagridy

V elementu **datagridViews** jsou obdobně jako u grafu definovány jednotlivé datagridy – jejich záložky (viz obr. 3.8). Každý datagrid se také stejným způsobem odkazuje na svůj zdrojový pohled – **viewId**, a na své parametry – **datagridStyleId**.

Kapitola 4

Průzkum existujících nástrojů

Následující kapitola se věnuje průzkumu existujících nástrojů, které se zabývají předpisem pro zpracování dat a jejich zobrazením. Pro průzkum byly vybrány nástroje, které řeší podobnou problematiku jako náš Jazyk, tedy předpis pro zpracování velkého množství dat do přehledné formy a především tvorbu těchto předpisů v GUI. Některé z těchto nástrojů mají navíc tento předpis ve vlastním doménově specifickém jazyce, podobně jako Předpisy v našem Jazyce. Po prozkoumání těchto nástrojů bych pak měl být schopen dělat lepší rozhodnutí při návrhu GUI pro náš Jazyk.

Zkoumané sofistikované komerční nástroje budou často rozsáhlé a budou řešit širší spektrum funkcionalit a v těchto případech se zaměřím především na věci podobné našemu Jazyku a Předpisům, mezi které patří zpracování časových řad, agregace, seskupování, filtrování, třídění a řazení dat. Dále na definici tabulek a spojnicových grafů, jejich parametrizaci (zde je zajímavé hlavně seskupování sloupců) a také na napojení těchto zobrazovacích komponent na zdrojová data (obdoba našich datových pohledů).

Ve všech nástrojích bude nakonec vyzkoušena tvorba předpisu pro zpracování dat podle dvou připravených příkladů. Oba tyto příklady budou obsahovat často používanou funkcionalitu datových pohledů, jako výběr, agregace, řazení a seskupování dat. Příklady budou podrobněji popsány až před jejich první realizací v prvním zkoumaném nástroji (podkapitoly 4.1.9 a 4.1.13).

Na konci kapitoly pak budou shrnuty a zhodnoceny poznatky z průzkumu těchto nástrojů.

4.1 Nástroj 1 – SSRS

Prvním zkoumaným nástrojem je nástroj SQL Server Reporting Services neboli SSRS. Jedná se o softwarový systém pro generování reportů (tedy zpráv) od společnosti Microsoft [2].

Systém umožňuje zobrazení reportů ve webovém prohlížeči, na mobilních zařízeních nebo jako obsah e-mailových zpráv. Systém běží on-premise – tedy

na hardware toho, kdo reporty tvoří. SSRS je součástí služeb pro relační databázový systém Microsoft SQL Server.

Reporty SSRS se dělí na tři typy:

1. tradiční reporty ve formě dokumentu, tzv. paginated reports – těmi se budu v průzkumu zabývat,
2. reporty pro mobilní zařízení (nejen pro Windows Phone) – umožňují přizpůsobení zobrazení na displej telefonu a tabletu,
3. reporty pro webové rozhraní – umožňují uživatelskou interakci s reportem – jsou generovány v HTML5 za dodržování standardů pro kompatibilitu s různými webovými prohlížeči.

V nástroji SSRS se dále řeší vzhled reportu, tedy rozmístění a nastavení zobrazovacích komponent zprávy atd. Náš Jazyk se tímto zabývá jen okrajově – rozmístění zobrazovacích komponent je pevně dáno šablonou aplikace. V Jazyce se pak pouze řeší kolik grafů a tabulek v šabloně bude a dále jim lze parametrizovat vzhled, ale pouze do omezené míry. Primárně se pak Jazyk věnuje zpracování dat, které se mají v zobrazovacích komponentách zobrazit.

4.1.1 Koncepty

Součástí SSRS je RDL – Report Definition Language. Jedná se o doménově specifický jazyk, který je založen na XML a slouží pro popis definice reportu. Definice reportu je tedy XML soubor s gramatikou tohoto jazyka a soubory tohoto typu mají příponu „.rdl“. Jazyk RDL popisuje propojení se zdroji dat, dotazy pro získání dat, výrazy, parametry reportu, obrázky, tabulky atd. [3].

Grafická uživatelská rozhraní pro tvorbu reportů v tomto jazyce jsou dvě – nástroj Report Builder a Report Designer. Report Designer je integrován ve vývojovém prostředí Microsoft Visual Studio, zatímco Report Builder je samostatná aplikace, která připomíná prostředí programů z balíčku Microsoft Office.

Hlavní koncepty v reportech:

- Zdroje dat (data sources) – připojení ke zdroji dat, nad kterými pak běží dotazy (queries). Umožňuje připojení k relační databázi, webové službě atd.
- Datasety (data sets) – sada dat, jako výsledek nějakého dotazu nad zdrojem dat. Samotná data nejsou součástí definice reportu. V definici datasetu jsou obsaženy dotazy, parametry, filtry atd.
- Parametry reportu (report parameters) – lze pomocí nich parametrizovat vzhled zprávy, určovat viditelnost prvků pro danou skupinu uživatelů atd.

- Prvky reportu (report items) – popisky, obrázky a další pomocné vzhledové elementy, které mohou být součástí zprávy.

- Datové regiony (data regions) – zobrazují data z právě jednoho datasetu. Umožňují běžné způsoby zobrazení dat – v tabulce, v matici (matrix), v seznamu, v grafu atd.

  - tabulka – počet sloupců je statický, počet řádků je dynamický. (Statický počet sloupců je značné omezení, neboť v našich Předpisech není často dopředu známý počet výsledných sloupců.) Data lze v tabulce seskupovat.
  - matice – dynamický počet sloupců i řádků. Sloupce i řádky mohou být do sebe vnořovány a tak lze data v matici seskupovat.
  - grafy – sloupcové, koláčové, spojnicové a další.

Definice reportu může být rozdělena na několik částí (report parts) do samostatných dokumentů. Tyto části pak lze znovupoužít v jiných reportech a tím omezit duplikaci kódu.

Definice reportu je zpracována pomocí report serveru. Report server nejprve zpracuje definici reportu a pak zprávu renderuje. Po zpracování definice vznikne meziformát (intermediate format), který je nezávislý na způsobu zobrazení (HTML, PDF atd.), a až při renderování je vykreslen daným způsobem.

4.1.2 Prostředí pro tvorbu reportů

Jak již bylo zmíněno, pro SSRS existují dvě prostředí pro tvorbu reportů: Report Builder a Report Designer. Obě tyto prostředí tvoří interně předpis v RDL jazyce [4].

Report Designer je součástí Business Intelligence Development Studio, což je nadstavba integrovaná do Microsoft Visual Studio. Pouze však do zastaralé verze Visual Studio 2008 a SQL Server 2008, dále již zřejmě není Business Intelligence Development Studio podporováno [5].

Report Builder je samostatná aplikace, která je součástí platformy Microsoft SQL Server. Report Builder umožňuje použít části již existujících reportů z Report Part Gallery. Podpora Report Builderu je až do současné verze Microsoft SQL Server 2016 [6].

Vzhledem k podpoře do současných verzí se zaměřuji na nástroj Report Builder.

Zprovoznění Report Builderu probíhalo bez problému. Nebylo ani nutné se zabývat platformou Microsoft SQL Server – stačilo pouze stáhnout samostatnou aplikaci Report Builder ze stránek výrobce (dostupná je však pouze anglická verze). Instalace probíhala standardně a bez komplikací a po instalaci již bylo vše připraveno k prvním spuštění nástroje.

4.1.3 Tvorba prvního reportu v Report Builder

Dále se budu zabývat prací s Report Builderem. Na první pohled je zřejmá podobnost šablony aplikace (pás karet v horní části obrazovky apod., viz obr. 4.1) s programy z balíčku Microsoft Office.

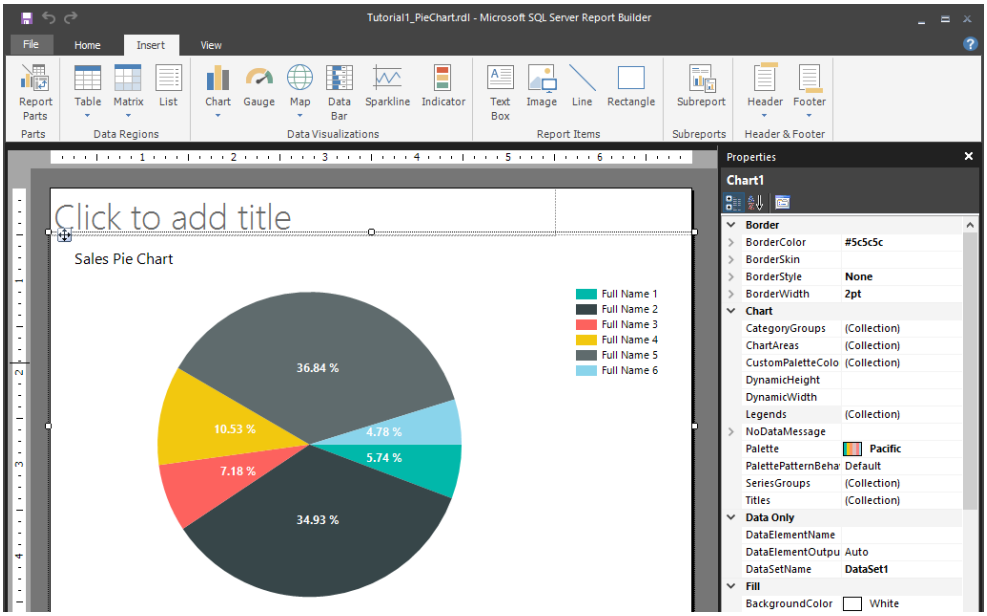
Pro seznámení s nástrojem jsem zvolil ze stránek výrobce tutoriál pro vytvoření jednoduchého reportu, která bude zobrazovat data v koláčovém grafu. Tento report je tzv. offline, což znamená, že pro něj nebude potřeba mít připravenou databázi, ze které by čerpal data, ale jeho zdrojová data budou umístěna přímo v reportu [7].

Ostatní tutoriály pak již typicky vyžadují přístup k SQL databázi [8].

Data tohoto offline reportu jsou součástí XML dotazu, který je následující:

```
<Query>
<ElementPath>
  Root /S  {@Sales (Integer)} /C  {@FullName}
</ElementPath>
<XmlData>
<Root>
<S Sales="150">
  <C FullName="Jae Pak" />
</S>
<S Sales="350">
  <C FullName="Jillian  Carson" />
</S>
<S Sales="250">
  <C FullName="Linda C Mitchell" />
</S>
<S Sales="500">
  <C FullName="Michael Blythe" />
</S>
<S Sales="450">
  <C FullName="Ranjit Varkey" />
</S>
</Root>
</XmlData>
</Query>
```

První fází je tvorba zdroje dat a datasetu, do kterého má být vložen zmíněný dotaz. Narážím však na zdánlivě nesmyslnou chybu, která brání tvorbě datasetu, znění chyby je: „.5in is not a valid unit designator“. Po hledání řešení na internetu jsem našel, že se jedná o problém s desetinnou čárkou / tečkou, který je způsoben tím, že mám českou verzi Windows, ale anglickou verzi Report Builderu. Po nastavení anglického formátování čísel ve Windows



Obrázek 4.1: Editace tutorialu v Report Builder

byla chyba vyřešena (v ovládacím panelu Oblast nastavení formátu z „nastavit jazyk podle nastavení Windows“ na angličtinu).

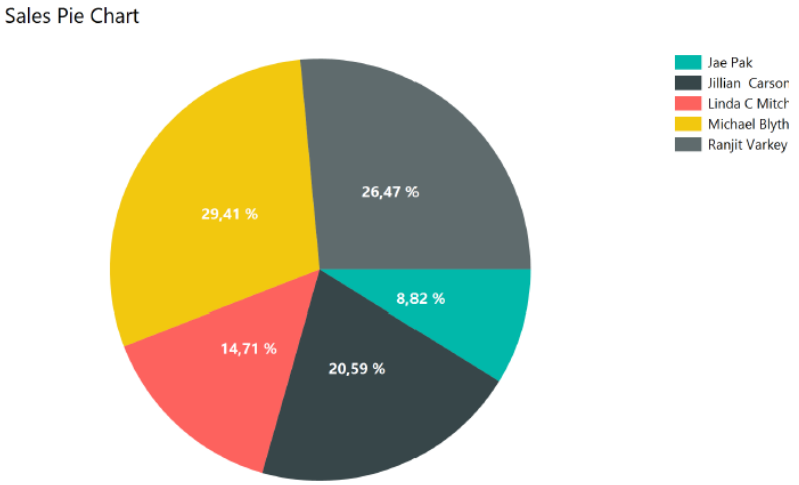
Po dokončení tvorby zdroje dat a datasetu v dialogovém okně se mi otvírá celé GUI, které obsahuje několik hlavních oken (viz obr. 4.1):

- Designer okno (dále Designer) – okno pro návrh celého reportu. (Toto okno je podobné jako Designer okno pro tvorbu Windows Forms nebo WPF aplikací v Microsoft Visual Studio.)

- Properties okno – okno pro nastavení parametrů označeného prvku v Designer oknu. (Toto okno se zdá shodné s Properties oknem v Microsoft Visual Studio.)

V Designeru je vidět, jak bude zpráva vypadat, nezávisle na konkrétních datech ve zdroji dat, ale pouze na jejich struktuře – je zde příkladový počet a hodnoty dat – Full Name 1 namísto Jae Pak apod. (viz obr. 4.1). Něco podobného by bylo zřejmě vhodné zobrazit při tvorbě datových pohledů v našem nástroji, protože v této fázi ještě také není přesná podoba vstupních dat známá.

Pro zobrazení výsledné podoby zprávy je třeba zprávu spustit – tlačítko Run, kde se připraví zpráva již s konkrétními daty (obr. 4.2). Výslednou podobu zprávy je možné exportovat do různých formátů – např. do PDF nebo do formátů z balíčku Microsoft Office (.docx, .xlsx, .pptx).



Obrázek 4.2: Výsledný koláčový graf z tutorialu

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <Report MustUnderstand="df"
3           xmlns="http://schemas.microsoft.com/sqlserver/reporting/2016/01/reportdefinition"
4           xmlns:rd="http://schemas.microsoft.com/SQLServer/reporting/reportdesigner"
5           xmlns:df="http://schemas.microsoft.com/sqlserver/reporting/2016/01/reportdefinition/defaultfontfamily">
6     <df:DefaultFontFamily>Segoe UI</df:DefaultFontFamily>
7     <AutoRefresh>0</AutoRefresh>
8     <DataSources>...</DataSources>
19    <DataSets>...</DataSets>
59    <ReportSections>...</ReportSections>
526   <ReportParametersLayout>...</ReportParametersLayout>
532   <rd:ReportUnitType>Inch</rd:ReportUnitType>
533   <rd:ReportID>5da33be6-f238-4d41-9e06-8ea11a4a6b5b</rd:ReportID>
534 </Report>
```

Obrázek 4.3: RDL předpis pro tutorial

4.1.4 Náhled do definice reportu

Uložení definice zprávy je do jednoho výstupního souboru typu „.rdl“ – tedy definice reportu v RDL jazyce.

Následuje náhled do RDL souboru (viz obr. 4.3, celý soubor se nachází na přiloženém DVD), který probíhá intuitivně se znalostí výše zmíněných konceptů jazyka a podrobnosti jsou pak dohledávány ze specifikace jazyka [9].

Na první pohled se jedná o obyčejný XML dokument, který je ale značně rozsáhlý – i pro tento jednoduchý report má soubor přes 500 řádek, proto se zaměřím pouze na relevantní prvky.

Kořenový element dokumentu se nazývá *Report* a obsahuje podelementy *DataSources* (zdroje dat), *DataSets* (datasety), *ReportSections* (sekce reportu) a *ReportParametersLayout* (další vzhledové parametry reportu).

Element se zdroji dat (viz obr. 4.4) obsahuje jeden zdroj dat (element *DataSource*). Zdroj dat obsahuje informace o připojení k datům: je zde název zdroje dat, jeho typ – v mém případě XML a také tzv. connection string pro připojení k databázi – v mém případě žádný, vzhledem k offline verzi

```
<DataSources>
  <DataSource Name="MyPieChart">
    <ConnectionProperties>
      <DataProvider>XML</DataProvider>
      <ConnectString />
      <IntegratedSecurity>true</IntegratedSecurity>
    </ConnectionProperties>
    <rd:SecurityType>Integrated</rd:SecurityType>
    <rd:DataSourceID>d7a9d82e-16f3-402a-800a-209d3046150d</rd:DataSourceID>
  </DataSource>
</DataSources>
```

Obrázek 4.4: Sekce DataSources v RDL pro tutorial

```
<DataSets>
  <DataSet Name="DataSet1">
    <Query>
      <DataSourceName>MyPieChart</DataSourceName>
      <CommandText>...</CommandText>
      <rd:UseGenericDesigner>true</rd:UseGenericDesigner>
    </Query>
    <Fields>
      <Field Name="Sales">
        <DataField>Sales</DataField>
        <rd:TypeName>System.Int32</rd:TypeName>
      </Field>
      <Field Name="FullName">
        <DataField>FullName</DataField>
        <rd:TypeName>System.String</rd:TypeName>
      </Field>
    </Fields>
  </DataSet>
</DataSets>
```

Obrázek 4.5: Sekce DataSets v RDL pro tutorial

reportu (viz výše). Všímám si podobnosti tohoto elementu s elementem Tables s popisem zdrojových tabulek v našem Předpise.

Element s datasety (viz obr. 4.5) obsahuje jeden dataset (element *DataSet*). Dataset obsahuje dotaz, ve kterém se nachází název zdroje dat (element *DataSourceName*) – ten slouží jako jeho identifikátor (obdoba atributu tableId v našem Předpise). Dále je součástí dotazu samotný dotazovací příkaz (element *CommandText*), který v mém případě zahrnuje i samotná data (viz offline report výše), což ale není typické – data obyčejně poskytuje zdroj dat. Všímám si podobnosti datasetů s našimi datovými pohledy.

Můj dotazovací příkaz (viz výše v podkapitole 4.1.3) nad zdrojem dat typu XML má formu XML fragmentu, který má pevně danou strukturu a obsahuje syntaxi podobnou jazyku XPath (dotazovací jazyk nad XML), použitou pro výběr sloupců dat apod. Typickými zdroji dat jsou však relační databáze a nad nimi jsou dotazovací příkazy v jazyce SQL [10].

Kromě dotazu obsahuje dataset ještě tzv. fieldy (fields). Field může být

```
59  <ReportSections>
60    <ReportSection>
61      <Body>
62        <ReportItems>
63          <Chart Name="Chart1">...</Chart>
433         <Textbox Name="ReportTitle">...</Textbox>
465       </ReportItems>
466       <Height>4.66181in</Height>
467       <Style>...</Style>
472     </Body>
473     <Width>7in</Width>
474     <Page>
475       <PageFooter>...</PageFooter>
518       <LeftMargin>1in</LeftMargin>
519       <RightMargin>1in</RightMargin>
520       <TopMargin>1in</TopMargin>
521       <BottomMargin>1in</BottomMargin>
522       <Style />
523     </Page>
524   </ReportSection>
525 </ReportSections>
```

Obrázek 4.6: Sekce ReportSections v RDL pro tutorial

buď jeden ze sloupců / řady hodnot, které jsou výsledkem dotazu – pak obsahuje element *DataField*, nebo může být definován výrazem – pak obsahuje element *Value*. Výraz může vypadat například následovně: „=Fields!Price.Value + Fields!Tax.Value“, který by značil součet řad Price a Tax. Tyto fieldy lze pak používat ve výrazech v různých částech dokumentu. Podrobněji budou výrazy popsány v následující podkapitole 4.1.5.

Element se sekcemi reportu (viz obr. 4.6) obsahuje jednu sekci (element *ReportSection*) s vlastnostmi stránky (element *Page*) a tělem reportu (element *Body*). V těle, až na různé styly, jsou prvky reportu (element *ReportItems*) – v mém případě je zde graf (element *Chart*).

Graf (viz obr. 4.7) kromě vzhledových parametrů, které pro návrh GUI naší aplikace nejsou příliš zajímavé, obsahuje napojení na data. V elementu *ChartDataPointValues* je určeno, odkud se mají brát do grafu hodnoty, v mém případě je zde výraz „=Sum(Fields!Sales.Value)“, značící, že se použije řada / field s názvem Sales. Dále jsou pomocí výrazů definovány popisky, které se mimo jiné zobrazují v legendě grafu (podelement *Label* elementu *ChartMembers*) s výrazem „=Fields!FullName.Value“, je tedy použita řada hodnot / field s názvem FullName.

Ruční tvorba těchto definic by zřejmě byla ještě méně udržitelná, než jak je to u našeho Předpisu.

```
63          <Chart Name="Chart1">
64            <ChartCategoryHierarchy>
65              <ChartMembers>
66                <ChartMember>
67                  <Group Name="Chart1_Category">...</Group>
72                  <SortExpressions>...</SortExpressions>
77                  <Label>=Fields!FullName.Value</Label>
78                </ChartMember>
79              </ChartMembers>
80            </ChartCategoryHierarchy>
81            <ChartSeriesHierarchy>...</ChartSeriesHierarchy>
88            <ChartData>...</ChartData>
135           <ChartAreas>...</ChartAreas>
367           <ChartLegends>...</ChartLegends>
388           <ChartTitles>...</ChartTitles>
402           <Palette>Pacific</Palette>
403           <ChartBorderSkin>...</ChartBorderSkin>
410           <ChartNoDataMessage Name="NoDataMessage">...</ChartNoDataMessage>
419           <DataSetName>DataSet1</DataSetName>
420           <Top>0.5in</Top>
421           <Height>4.16181in</Height>
422           <Width>7in</Width>
423           <Style>...</Style>
432         </Chart>
```

Obrázek 4.7: Sekce Chart v RDL pro tutorial

4.1.5 Výrazy

Po prohlédnutí témat ostatních tutoriálů usuzuji, že se příliš nevěnují částem, které by byly relevantní pro srovnání s naším Jazykem, proto budu dále nástroj zkoumat bez tutoriálů [11]. Všiml jsem si ale, že se v ostatních tutoriálech často používají výrazy, proto je zde trochu více prozkoumám.

Výrazy jsou používány napříč definicí reportu pro výpočet hodnot parametrů, v dotazech atd. Jsou textové povahy, ale v Report Builderu je pro ně editační okénko Expression dialog box (viz obr. 4.9), které jejich tvorbu usnadňuje [12].

Tyto výrazy jsou psány v jazyce Microsoft Visual Basic a mohou obsahovat odkazy na fieldy definované v datasetech. Dále výrazy mohou obsahovat aritmetické operátory, práci s kolekcemi a volání různých funkcí (součet, průměr, ale i funkce nad řetězci, daty apod.).

Pro lepší porozumění následuje několik ukázkových výrazů [13]:

- „=Round(1.3 * 5) / 5“ – obsahuje konstanty, aritmetické operace a funkci zaokrouhlení,
- „=Today()“ – vrací aktuální datum,
- „=Year(Fields!OrderDate.Value)“ – složený výraz obsahující identifikaci fieldu OrderDate z datasetu.

Tabulka 4.1: Data k tutorialu SSRS

Full Name	Sales
Jae Pak	150
Jillian Carson	350
Linda C Mitchell	250
Michael Blythe	500
Ranjit Varkey	450

4.1.6 Spojnicové grafy a tabulky – vstupní data prvního reportu

Dále si v nástroji vyzkouším tvorbu spojnicových grafů a tabulek, které lze definovat i v našem Jazyce. Pro zkoušení použiji data z prvního tutoriálu – viz tabulka 4.1.

Uvažuji první sloupec jako indexační (u časových řad je indexační sloupec datum, resp. časová osa) a druhý sloupec jako datový (stejně jako u časových řad).

Přidám do reportu prázdný graf a v Designeru nastavím napojení dat pomocí následujících parametrů grafu. Parametr Values nastavím na field (/ sloupec tabulky) Sales a parametr Category Groups nastavím na field FullName. Tím dostávám graf, který má na jednu řadu hodnot, na ose X jsou jména z tabulky a na ose Y jsou hodnoty.

I když v konceptech výše je rozlišena tabulka a matice, zdá se, že se jedná o jedno a to samé. Je dokonce vidět v Properties okně, že přidaný prvek se jmenuje tablix, tedy kombinace názvů table a matrix. Podobným způsobem jako graf přidávám tedy i tabulku.

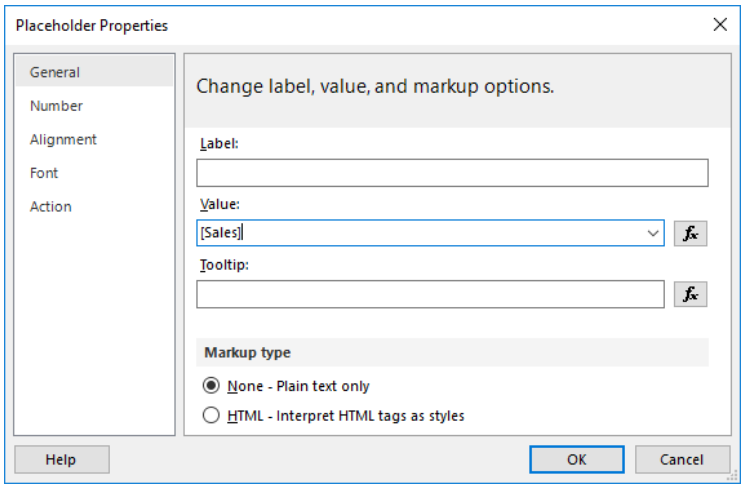
Když se snažím editovat parametry popisující vybraná data, zjišťuji, že jsou to ve skutečnosti výrazy, které lze manuálně upravit (viz obr. 4.8 a 4.9).

Když dojde při manuální úpravě výrazu k překlepu, chybu se dozvím až při zobrazení výsledné zprávy, kde software vypíše chybu a výslednou zprávu nezobrazí (viz obr. 4.10). Správnost výrazu tedy není kontrolována již při jeho návrhu.

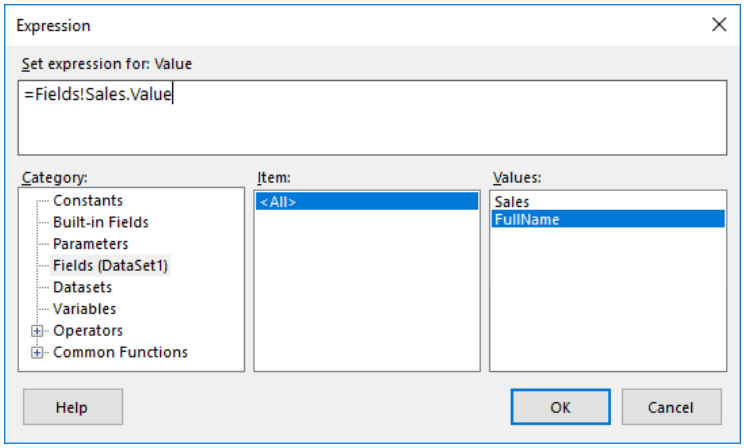
U grafů i tabulek se mi jako výraz popisující napojení na data automaticky tvoří výraz „Sum(Sales)“, přitom moje data jsou tvořena jednou řadou hodnot a žádné agregace nejsou potřeba. Přítomnost součtu ale má význam, když jsou zdrojová data seskupována – viz 4.1.13 dále. A skutečně, když pro tato data ručně edituji výraz a součet odstraním, report stále funguje. Podařilo se mi tedy zobrazit jeden sloupec dat v tabulce a v grafu.

4.1.7 Spojnicové grafy a tabulky – rozšíření vstupních dat

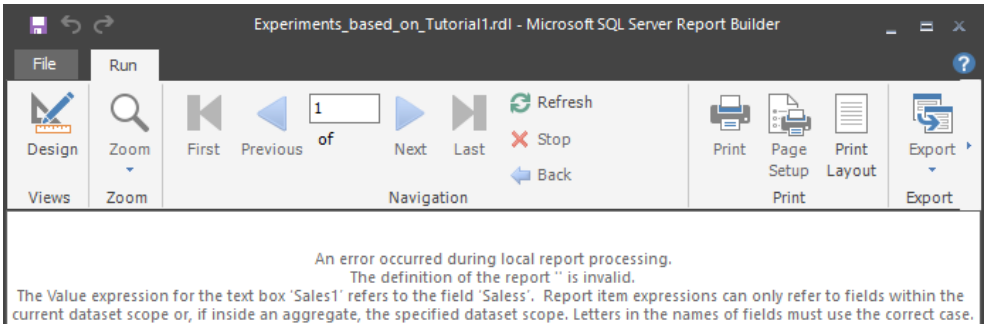
Dále se zaměřím na zobrazení několika řad hodnot se společnou indexací datem, což bude bližší k zobrazení časových řad, kterým se zabývá náš Předpis.



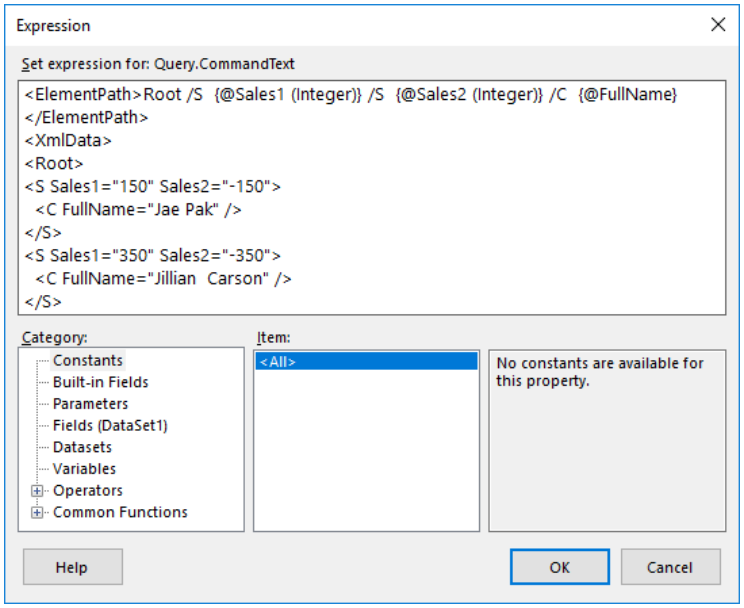
Obrázek 4.8: Ruční editace výrazu



Obrázek 4.9: Editace výrazu v Expression Editoru



Obrázek 4.10: Spuštění reportu s chybným výrazem

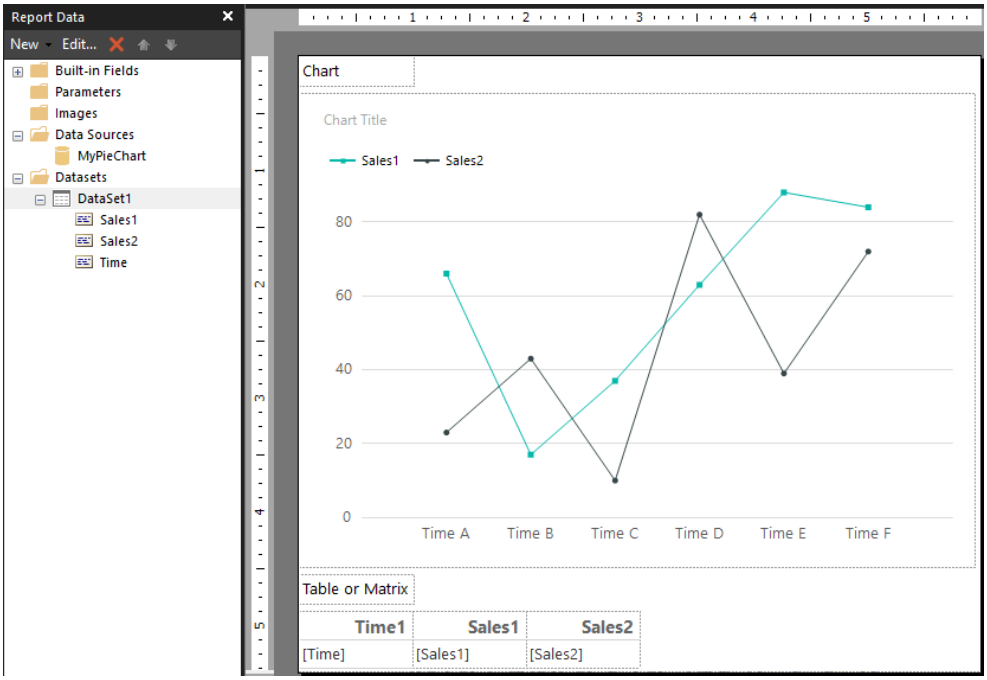


Obrázek 4.11: Editace XML dotazu v Expression Editoru

Musím tedy nejprve upravit zdrojová data, resp. dotaz (offline dotaz obsahuje data). Pro editaci dotazu se použije opět okno pro editaci výrazů. Toto okno mi však pro editací dotazu nenabízí žádná usnadnění oproti textovému editoru a okno dokonce nemá ani scrollbar (viz obr. 4.11). Je to tím, že dotaz je v podobě XML, pro SQL dotazy je editace výrazů v okně přívětivější (viz dále).

Po úpravě dotazu a jeho zjednodušení mám připravena data dvou hodinových časových řad Sales1 a Sales2. Dotaz nyní vypadá následovně (původní podoba dotazu viz podkapitola 4.1.3 výše):

```
<Query>
<ElementPath>
  Root /S {@Sales1 (Integer), @Sales2 (Integer), @Time}
</ElementPath>
<XmlData>
<Root>
<S Sales1="150" Sales2="15" Time="01:00" />
<S Sales1="350" Sales2="35" Time="02:00" />
<S Sales1="250" Sales2="25" Time="03:00" />
<S Sales1="500" Sales2="50" Time="04:00" />
<S Sales1="450" Sales2="45" Time="05:00" />
</Root>
</XmlData>
</Query>
```



Obrázek 4.12: Graf a tabulka v Designer režimu Report Builderu

Na obrázcích 4.12 a 4.13 je vidět report po napojení nového dotazu na graf a tabulku.

Zobrazovací komponenty jde vložit buď prázdné a nakonfigurovat si je v Designeru a pomocí Properties okna, nebo je možné použít průvodce (Wizard). Průvodce je formulář, který má několik kroků, ve kterých je vyplněno napojení na data a základní parametrizace komponenty.

Napojení grafu na data v průvodci vypadá následovně (viz obr. 4.14). Jsou zde dostupné fieldy ze zdrojového datasetu, které lze přiřazovat do kategorií (Categories) – u spojnicového grafu odpovídají ose X, jako hodnoty (Values) – odpovídají řadám na ose Y. Parametr Series souvisí se seskupováním sloupců dat a bude diskutován až v podkapitole 4.1.13.

Toto napojení dat do grafu v tomto příkladě v podstatě realizuje funkci datového pohledu pro výběr v našem Jazyce. V našem Jazyce se grafu přiděluje již hotový datový pohled, který se zobrazí celý, je tedy nutné provést výběr časových řad již v datových pohledech.

4.1.8 Query Designer

V SSRS existují různé nástroje pro tvorbu dotazů nad datasety. Některé tyto nástroje umožňují volbu mezi vizuálním módem a přímým náhledem na dotazovací jazyk (editace kódu) [14].

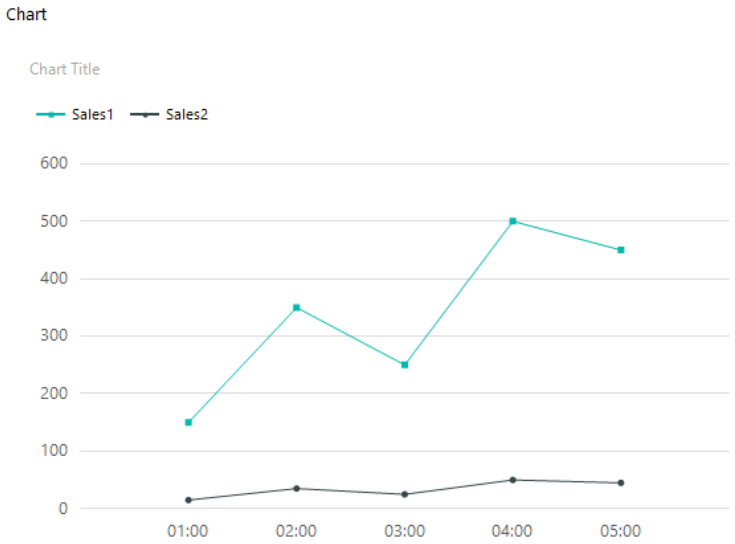
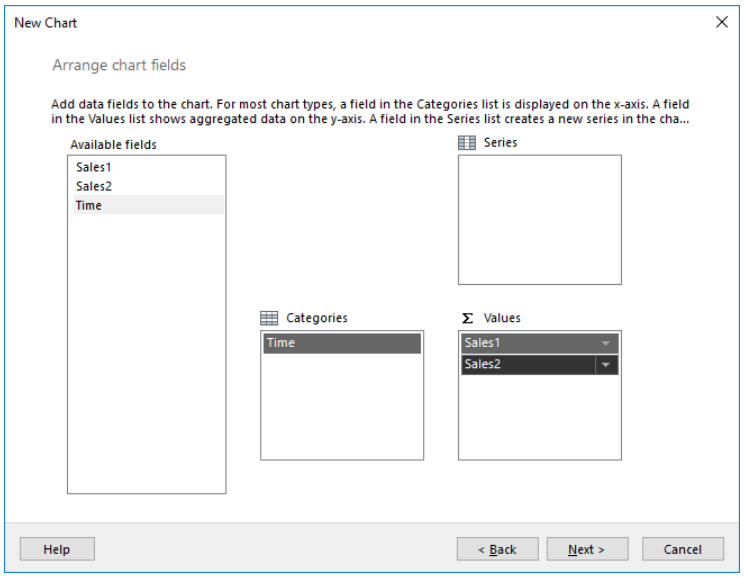


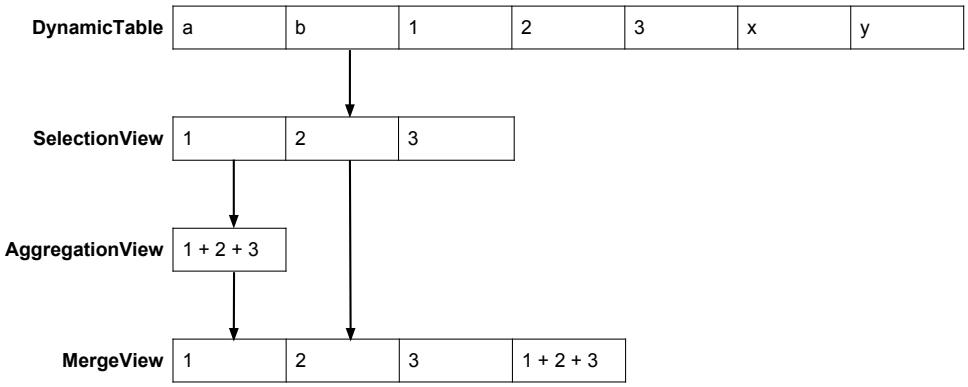
Table or Matrix

Time1	Sales1	Sales2
01:00	150	15
02:00	350	35
03:00	250	25
04:00	500	50
05:00	450	45

Obrázek 4.13: Graf a tabulka v Run režimu Report Builderu



Obrázek 4.14: Průvodce pro napojení dat do grafu



Obrázek 4.15: Strom datových pohledů k příkladu 1

Query Designer je nástroj pro tvorbu dotazů v Report Builderu. Pro dotazy v XML Query Designer vizuální mód nenabízí, pouze umožní vyzkoušet jejich vyhodnocení. Nicméně nástroj nabízí vizuální mód pro tvorbu dotazů nad Microsoft SQL databází [15].

Musím tedy nejprve zprovoznit tuto databázi. Nejprve jsem stáhnul a nainstaloval Microsoft SQL Server 2016 Express a dále SQL Server Management Studio, tedy prostředí pro správu této databáze. Instalace probíhá bez větších problémů. Jakmile je oboje nainstalováno, pomocí SQL Server Management Studia zakládám testovací databázi s testovací tabulkou dat.

Pro napojení databáze do Report Builderu je potřeba přidat nový zdroj dat s connection stringem pro připojení k mé testovací databázi, tedy „Data Source=localhost\\SQLEXPRESS; Initial Catalog=test_db“. Dále lze přidat dataset, který tvoří SQL dotaz nad tímto zdrojem dat v nástroji Query Designer.

Pro otestování Query Designeru a další funkcionality Report Builderu jsem si připravil dva testovací příklady. Tyto testovací příklady mají za úkol snažit se nasimulovat funkcionalitu datových pohledů našeho Jazyka ve zkoumaném nástroji. Příklady obsahují operace s daty, které umí datové pohledy a jejich skládání. První příklad testuje jednodušší a druhý složitější funkcionalitu. Pro každý příklad bude potřeba připravit vlastní testovací sadu dat.

4.1.9 Příklad na datové operace 1

První příklad se zabývá výběrem časových řad, jednoduchými agregacemi a sloučením vybraných časových řad se vzniklými agregovanými řadami.

Jedná se o již použitý příklad v kapitole 3 a pro zopakování je tento příklad znovu uveden následujícím odstavcem.

Cílem stromu datových pohledů (viz obr. 4.15) je vybrat určité časové řady a zobrazit k nim jejich součet. Ve zdrojové tabulce jsou i časové řady, které nechceme zobrazit, proto pomocí pohledu pro výběr vybereme pouze

Tabulka 4.2: Ukázka dat k prvnímu srovnávacímu příkladu

date	a	b	one	two	three	x	y
1. 1. 2000, 0:00	10	20	1	2	3	30	40
1. 1. 2000, 1:00	10	20	1	2	3	30	40
1. 1. 2000, 2:00	10	20	1	2	3	30	40
1. 1. 2000, 3:00	10	20	1	2	3	30	40
1. 1. 2000, 4:00	10	20	1	2	3	30	40

potřebné časové řady. Dále chceme zobrazit součet vybraných časových řad, připravíme si jej tedy pomocí pohledu pro agregace. Ve výsledném zobrazení chceme mít jak vybrané časové řady, tak řadu s jejich součtem, sloučíme tedy datové pohled pro výběr a pohled pro agregace. Tím dostáváme cílový pohled, který lze například zobrazit v datagridu.

Nejprve je potřeba připravit si testovací sadu dat, tedy testovací SQL tabulku.

Tabulka bude obsahovat data pro časové řady. Časové řady mají společnou časovou osu, pro příklad hodinovou pro jeden den (1. 1. 2000). Tato společná osa bude prvním sloupcem tabulky. Pro časové řady z příkladu tedy A, B, 1, 2, 3, X, Y potřebuji ještě jejich hodnoty. Každá časová řada bude mít tedy v tabulce vlastní sloupec s hodnotami. Časové řady tedy budou identifikovány dle názvu sloupce v SQL tabulce. Ukázku dat naleznete v tabulce 4.2.

Co se týče datových typů sloupců, pro data stačí přesnost na hodiny, zvolím tedy postačující typ *smalldatetime* [16]. Hodnoty časových řad jsou reálná čísla bez speciálního požadavku na přesnost, volím tedy datový typ *real* [17].

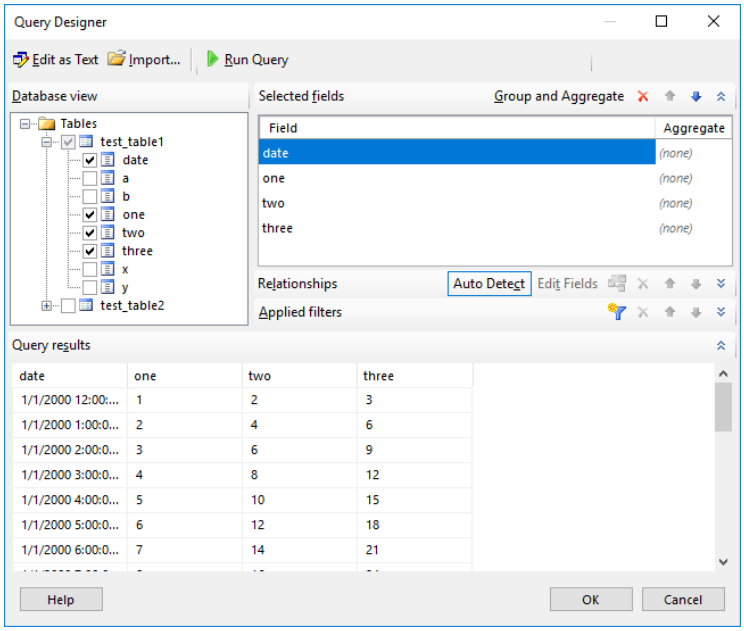
Předpokládaný dotaz nad tabulkou, který mi z tabulky vybere požadovaná data, by tedy měl být výběr sloupců 1, 2, 3 pomocí klauzule SELECT. Klauzule SELECT dále umí i pracovat s aritmetickými výrazy, mohla by tedy obsahovat výraz (1 + 2 + 3), který zajistí sečtení sloupců po jednotlivých řádcích [18].

4.1.10 Příklad na datové operace 1 – Query Designer

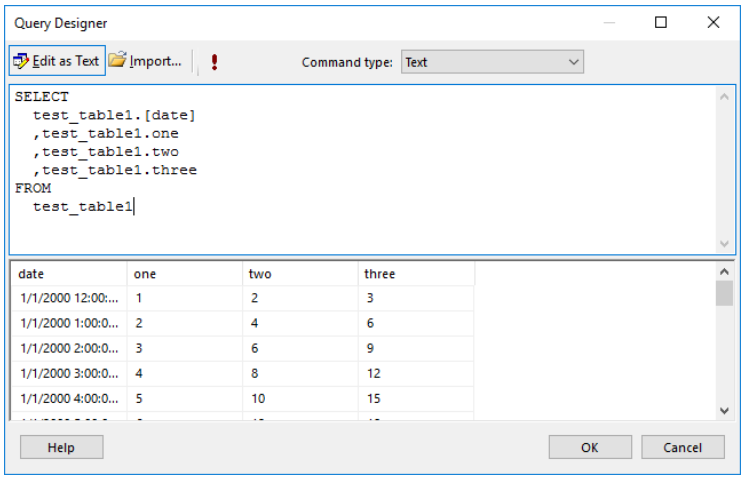
V reportu mám založený zdroj dat nad testovací databází (zdroj dat je celá databáze nikoli pouze tabulka). Dále budu tvořit dataset, pro který se pokusím v nástroji Query Designer vytvořit požadovaný SQL dotaz.

V Query Designeru lze přepínat mezi vizuálním módem a textovým náhledem na SQL dotaz (viz obr. 4.16 a 4.17, přepínání módu pomocí tlačítka „Edit as Text“ vlevo nahoře).

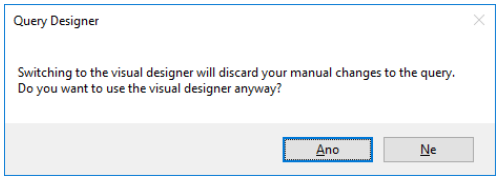
Ve vizuálním módu vyberu sloupce 1, 2, 3 s časovou osou (levé horní podokno), nelze ale přidat nový sloupec jako součet existujících. Zřejmě se tyto agregace neřeší v datasetu (ten pouze vybere, jaká data jsou potřeba), ale až při napojení dat do zobrazovací komponenty. Query Designer mi v tuto



Obrázek 4.16: Query Designer – Design View



Obrázek 4.17: Query Designer – Text View



Obrázek 4.18: Ztráta změn při ruční editaci dotazu

chvíli nabízí z tabulky pouze vybrat sloupce a případně filtrovat řádky pomocí podmínky, což ale v tuto chvíli nepotřebuji.

Když zkouším upravit SQL dotaz manuálně, tak jsem nucen před přepnutím do vizuálního módu zahodit provedené změny (viz obr. 4.18).

Nicméně při manuální úpravě dotazu s využitím aritmetické operace v klauzuli SELECT dosahuji kýženého výsledku – dotaz pak vypadá následovně:

```
SELECT
  test_table1.[date]
  ,test_table1.one
  ,test_table1.two
  ,test_table1.three
  ,(test_table1.one + test_table1.two + test_table1.three) AS sum
FROM
  test_table1
```

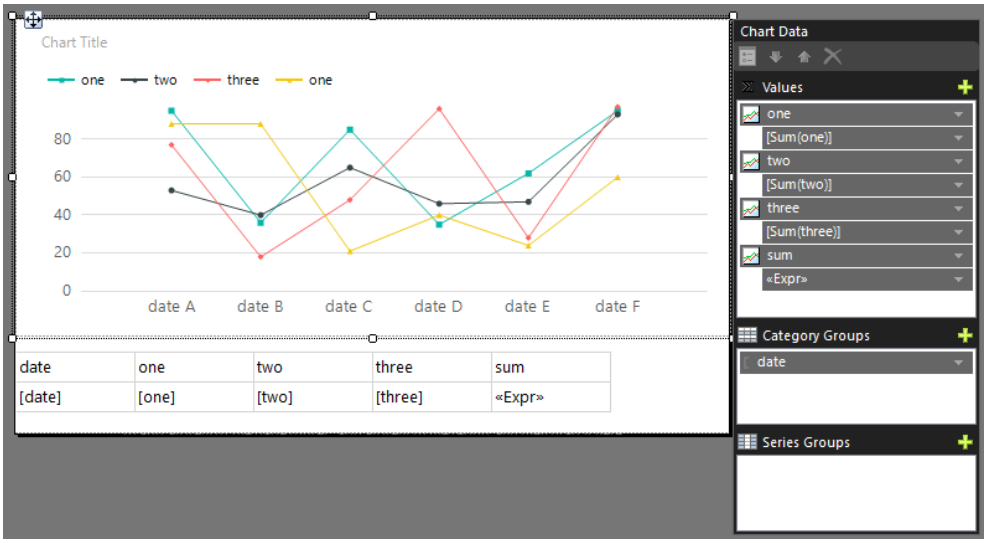
Manuální úprava v kódu mě ale pro návrh GUI příliš nezajímá – zkusím tedy najít jiný způsob jak v Report Builderu (v GUI) provést nad daty součet.

Daří se mi to pomocí editace výrazu při výběru dat do zobrazovací komponenty (viz obr. 4.19 – nastavení grafu v pravé části obrazovky a nastavení tabulky přímo v jejich buňkách). U tabulky jsou sloupce definovány výrazem „=Fields!one.Value“ pro první řadu atd. (viz popis výrazů výše). Když přidám další sloupec s výrazem „=Fields!two.Value + Fields!three.Value + Fields!one.Value“ mám požadovaný výsledek. Obdobně pak pro graf. Report je tedy hotový (viz obr. 4.20).

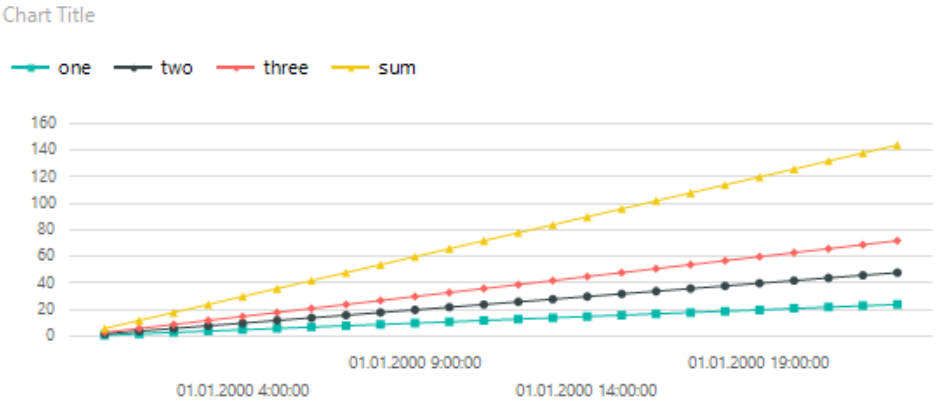
4.1.11 Příklad na datové operace 1 – náhled do definice reportu

Následuje náhled do definice vytvořeného reportu v RDL (viz obr. 4.21). Celý report je opět velmi rozsáhlý, má celkem 940 řádků, zaměřím se z něj tedy pouze na zajímavé věci v kontextu operací s daty (přeskočím tedy popisy vzhledu, rozmístění apod.). Celý RDL dokument se nachází na přiloženém DVD.

Element se zdroji dat (*DataSources*, viz obr. 4.22) obsahuje můj zdroj dat s connection stringem a s nastavením typu zdroje na SQL.



Obrázek 4.19: Report prvního příkladu v režimu Designer



date	one	two	three	sum
01.01.2000 0:00:00	1	2	3	6
01.01.2000 1:00:00	2	4	6	12
01.01.2000 2:00:00	3	6	9	18

Obrázek 4.20: Report prvního příkladu v režimu Run

```
<Report MustUnderstand="df"
        xmlns="http://schemas.microsoft.com/sqlserver/reporting/2016/01/reportdefinition"
        xmlns:rd="http://schemas.microsoft.com/SQLServer/reporting/reportdesigner"
        xmlns:df="http://schemas.microsoft.com/sqlserver/reporting/2016/01/reportdefinition/defaultfontfamily">
  <df:DefaultFontFamily>Segoe UI</df:DefaultFontFamily>
  <AutoRefresh>0</AutoRefresh>
  <DataSources>...</DataSources>
  <DataSets>...</DataSets>
  <ReportSections>...</ReportSections>
  <ReportParametersLayout>...</ReportParametersLayout>
  <rd:ReportUnitType>Inch</rd:ReportUnitType>
  <rd:ReportID>1bca668a-edbe-4dbe-a06c-e725753642b7</rd:ReportID>
</Report>
```

Obrázek 4.21: RDL předpis prvního příkladu

```
<DataSources>
  <DataSource Name="DataSource1">
    <ConnectionProperties>
      <DataProvider>SQL</DataProvider>
      <ConnectString>Data Source=localhost\SQLEXPRESS;Initial Catalog=test_db</ConnectString>
      <IntegratedSecurity>true</IntegratedSecurity>
    </ConnectionProperties>
    <rd:SecurityType>Integrated</rd:SecurityType>
    <rd:DataSourceID>bb6459d7-b2cd-48e8-b233-17453a9d68dd</rd:DataSourceID>
  </DataSource>
</DataSources>
```

Obrázek 4.22: Sekce DataSources v RDL prvního příkladu

Element s popisem datasetů (*DataSets*, viz obr. 4.23) obsahuje můj dataset. Dataset obsahuje element *Query*, kde se nachází samotný SQL dotaz a odkaz na zdroj dat (jeho název). Dále je zde element *Fields* s názvy a datovými typy vybraných fieldů, díky kterým lze pak sloupce používat ve výrazech (viz rozbor prvního RDL výše).

Element *ReportSections* (viz obr. 4.24) je nejrozsáhlejší – cca 850 řádek, většina z nich je ale popis vzhledu a stylů. Uvnitř se nachází popis grafu a tabulky.

Je velmi obtížné se vyznat se ve velkém množství elementů a najít co hledám – tedy práci zobrazovacích komponent s datasetem. Je zde totiž mnoho prázdných elementů a dále je většina elementů hluboce zanořená (hloubka je až v desítkách zanoření), přitom po několika zanořeních je zde často jen jeden nebo několik málo elementů. Definice je pak zbytečně složitá, i když popsaná zpráva složitá není a člověk se musí naučit ignorovat spoustu „šumu“ v definici, aby našel to co je podstatné (viz obr. 4.25).

V popisu tabulky nacházím napojení na dataset v elementu DataSetName. Definice zobrazených dat v buňkách je pak pomocí výrazu, který pouze vybírá příslušné fieldy (element Value, téměř 20 zanoření od kořene XML, viz obr. 4.25).

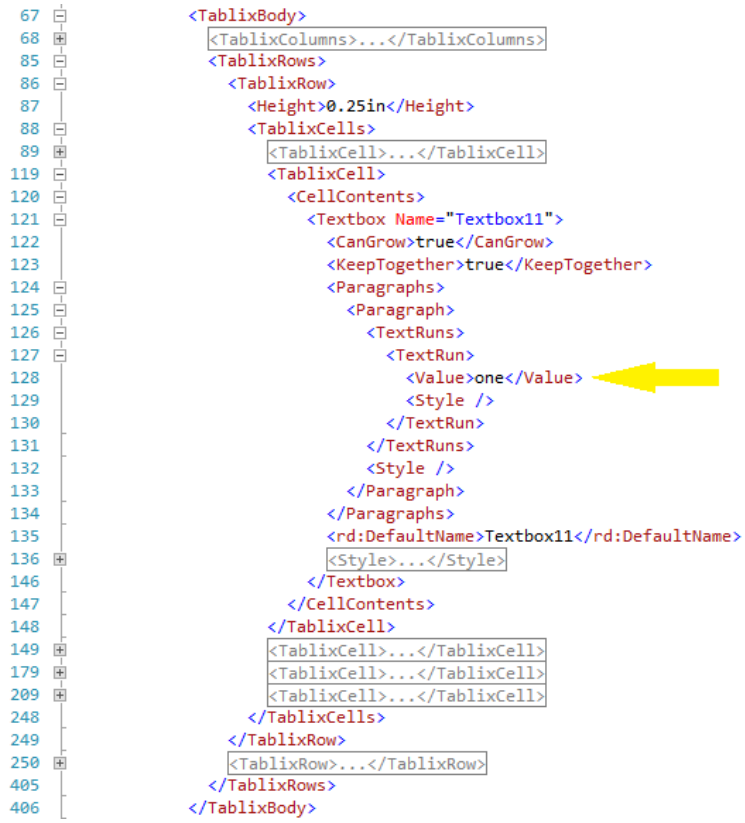
U grafu je napojení velmi podobné, je zde také element DataSetName a dále jsou zde elementy ChartDataPointValues obsahující výrazy pro výběr fieldů.

```
19  <DataSets>
20    <DataSet Name="DataSet1">
21      <Query>
22        <DataSourceName>DataSource1</DataSourceName>
23        <CommandText>SELECT
24    test_table1.[date]
25    ,test_table1.one
26    ,test_table1.two
27    ,test_table1.three
28  FROM
29    test_table1</CommandText>
30          <rd:DesignerState>...</rd:DesignerState>
40          <rd:UseGenericDesigner>true</rd:UseGenericDesigner>
41        </Query>
42        <Fields>
43          <Field Name="date">
44            <DataField>date</DataField>
45            <rd:TypeName>System.DateTime</rd:TypeName>
46          </Field>
47          <Field Name="two">
48            <DataField>two</DataField>
49            <rd:TypeName>System.Single</rd:TypeName>
50          </Field>
51          <Field Name="three">
52            <DataField>three</DataField>
53            <rd:TypeName>System.Single</rd:TypeName>
54          </Field>
55          <Field Name="one">
56            <DataField>one</DataField>
57            <rd:TypeName>System.Single</rd:TypeName>
58          </Field>
59        </Fields>
60      </DataSet>
61    </DataSets>
```

Obrázek 4.23: Sekce DataSets v RDL prvního příkladu

```
62  <ReportSections>
63    <ReportSection>
64      <Body>
65        <ReportItems>
66          <Tablix Name="Tablix2">...</Tablix>
436         <Chart Name="Chart1">...</Chart>
909       </ReportItems>
910       <Height>3.46875in</Height>
911       <Style>...</Style>
916     </Body>
917     <Width>6in</Width>
918     <Page>...</Page>
935   </ReportSection>
936 </ReportSections>
```

Obrázek 4.24: Sekce ReportSections v RDL prvního příkladu



Obrázek 4.25: Ukázka hloubky zanoření elementů

4.1.12 Příklad na datové operace 1 – shrnutí

To co u mě realizoval datový pohled pro výběr, se v nástroji provedlo zaškrtáním sloupců v Query Designeru, který vytvořil příslušný SQL dotaz.

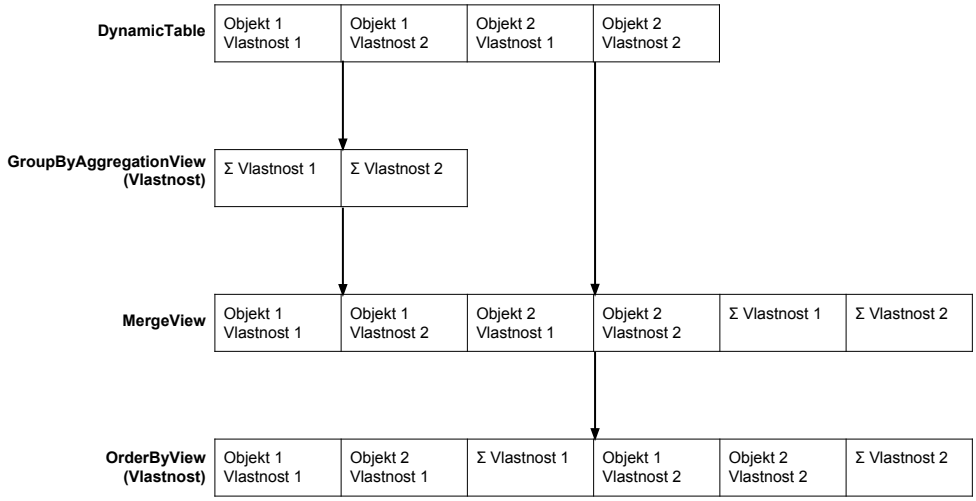
Datový pohled pro agregaci – pro součet sloupců pak byl vytvořen až v grafu / v tabulce pomocí aritmetické operace ve výrazu.

Funkce datového pohledu pro sloučení dat se přirovnává hůře, neboť výsledné sloupce bylo potřeba vybrat až v zobrazovací komponentě.

Ještě jednodušší mohl být postup, že bych při tvorbě datasetu nechal pomocí Query Designeru vybranou celou tabulku a SQL dotazem bych se tak vůbec nezabýval a sloupce bych vybral až v zobrazovací komponentě – vybrat jaký sloupec má být v jakém výsledném v jakém sloupci zobrazené tabulky / řady grafu a definovat sloupec součtu pomocí výrazu.

4.1.13 Příklad na datové operace 2

Druhý příklad je trochu složitější než první. Zaměřuje se na agregace se seskupením a řazení časových řad.



Obrázek 4.26: Strom datových pohledů k příkladu 2

Tento příklad pochází z mé bakalářské práce, kde jsem na něm demonstroval funkce datových pohledů. Následuje popis příkladu, který je převzatý z podkapitoly „Příklad struktury pohledů s využitím seskupení a řazení“ z mé bakalářské práce [1].

Pro lepší představu uvedu příklad struktury datových pohledů, kde se využívá seskupení a řazení (viz obr. 4.26). Cílem je získat tabulku seřazenou dle vlastností, kde u každé vlastnosti bude i její součet. Předpokládejme, že počet, ani název vlastností není dopředu znám, tudíž je nelze postupně vybrat. Nejdříve pomocí pohledu pro agregace se seskupením vytvořím součty jednotlivých vlastností, které sloučím se zdrojovou tabulkou. Nyní mám všechny sloupce, které potřebuji, pouze v nesprávném pořadí, pomocí pohledu pro řazení je tedy seřadím a získávám tak požadovanou tabulku.

Začnu tedy opět tvorbu testovací sady dat. Budu potřebovat časovým řadám přiřadit vlastnosti. Mapování na relační databázi už není v tomto případě přímočaré jako u prvního příkladu. Nejjednodušší datový model, ke kterému jsem došel je následující.

Vše je opět v jedné SQL tabulce, která má čtyři sloupce:

1. název časové řady (v obr. 4.26 by to byla kombinace názvu datového objektu a vlastnosti),
2. kategorie časové řady (v příkladu hodnota vlastnosti datového objektu),
3. datum – hodinová denní časová osa jako v prvním příkladě,
4. hodnota – reálná čísla jako v prvním příkladě.

Ukázka testovacích dat se nachází v tabulce 4.3.

Tabulka 4.3: Ukázka dat k druhému srovnávacímu příkladu

series	series_category	date	value
one	1	1. 1. 2000, 0:00	1
one	1	1. 1. 2000, 1:00	2
one	1	1. 1. 2000, 2:00	3
one	1	1. 1. 2000, 3:00	4
two	2	1. 1. 2000, 0:00	2
two	2	1. 1. 2000, 1:00	4
two	2	1. 1. 2000, 2:00	6
two	2	1. 1. 2000, 3:00	8
three	1	1. 1. 2000, 0:00	3
three	1	1. 1. 2000, 1:00	6
three	1	1. 1. 2000, 2:00	9
three	1	1. 1. 2000, 3:00	12

Jak je vidět, kombinace názvu časové řady a data tvoří unikátní identifikaci řádku (mohlo by se tedy jednat o primární klíč).

V datech tabulky existuje redundance, je zde duplikace přiřazení kategorie časovým řadám, což by při špatném vyplnění hodnot mohlo způsobit nekonzistence dat (jedna časová řada s více kategoriemi). Navzdory tomu, je však toto řešení upřednostněno před umístěním této informace do samostatné tabulky, kvůli jednoduššímu způsobu dotazování bez nutnosti spojování tabulek (klauzule JOIN).

4.1.14 Příklad na datové operace 2 – Úvaha o SQL dotazu

Pro jednoduchost dotazu neuvažuji neznámou množinu kategorií (dynamický počet a pojmenování vlastností datových objektů), ale pevně danou množinu kategorií 1, 2 . Výsledek dotazu bude sloučení (UNION) časových řad součet kategorie 1, součet kategorie 2 a jednotlivých časových řad.

Jednotlivou časovou řadu mohu získat dotazem, který vyfiltruje řádky pomocí názvu časové řady – tedy klauzule WHERE s podmínkou „series = 'XYZ'“.

Součet časové řady 1 (případně 2) získám následujícím způsobem. Nejprve vyfiltruji pouze řádky z kategorie 1 (WHERE kategorie = 1). Poté provedu agregaci pomocí klauzule GROUP BY – seskupím hodnoty podle sloupce „date“ a v těchto skupinách provedu součet sloupce „value“. Tzn., skupiny budou vždy obsahovat všechny hodnoty pro danou hodinu časové osy a pro tyto hodnoty provedu agregaci součet:

```
SELECT series_category, date, SUM(value) AS value
FROM table
WHERE series_category = '1' GROUP BY date
```

```
SELECT *
    FROM dbo.test_table2
    WHERE date < '2000-01-01 02:00:00'

UNION

SELECT CONVERT(varchar, series_category) + ' total' AS series, series_category, date, SUM(value) AS value
    FROM dbo.test_table2
    WHERE date < '2000-01-01 02:00:00'
    GROUP BY date, series_category

ORDER BY series_category
```

100 %

Results | Messages

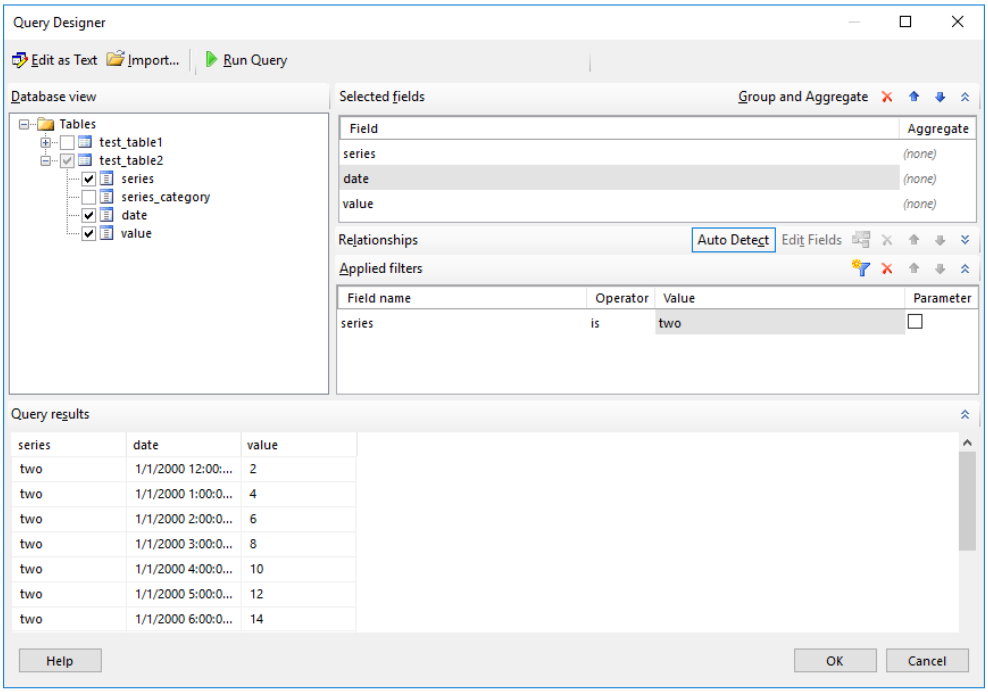
	series	series_category	date	value
1	1 total	1	2000-01-01 00:00:00	4
2	1 total	1	2000-01-01 01:00:00	8
3	one	1	2000-01-01 00:00:00	1
4	one	1	2000-01-01 01:00:00	2
5	three	1	2000-01-01 00:00:00	3
6	three	1	2000-01-01 01:00:00	6
7	2 total	2	2000-01-01 00:00:00	2
8	2 total	2	2000-01-01 01:00:00	4
9	two	2	2000-01-01 00:00:00	2
10	two	2	2000-01-01 01:00:00	4

Obrázek 4.27: SQL dotaz k druhému příkladu

Výše uvedené tři odstavce tedy dávají za vznik složenému SQL dotazu.

Pokud bych chtěl obecné řešení pro neznámou množinu kategorií, výsledný dotaz spolu s výsledkem odzkoušeným v Microsoft SQL Management Studio by vypadal následovně (viz obr. 4.27, aby nebyl výstup dotazu příliš dlouhý, použil jsem klauzule WHERE pro omezení 24-hodinových řad na první dvě hodiny):

- První SELECT vybere jednotlivé časové řady.
- druhý SELECT vybere součty časových řad po kategoriích:
  - seskupení provedu podle data a kategorie,
  - agregace je typu součet (SUM) a agregují se hodnoty časových řad (value),
  - abych mohl provést UNION, potřebuji mít sloupce ve stejném formátu – odpovídající počet a datové typy – do této tabulky tedy musím přidat sloupec series s názvem časové řady zde vytvořím spojením hodnoty kategorie s řetězcem „total“.
- UNION tyto časové řady sloučí do jedné tabulky.
- ORDER BY zajistí seřazení časových řad podle kategorie.



Obrázek 4.28: Query Designer – výběr jedné řady v druhém příkladu

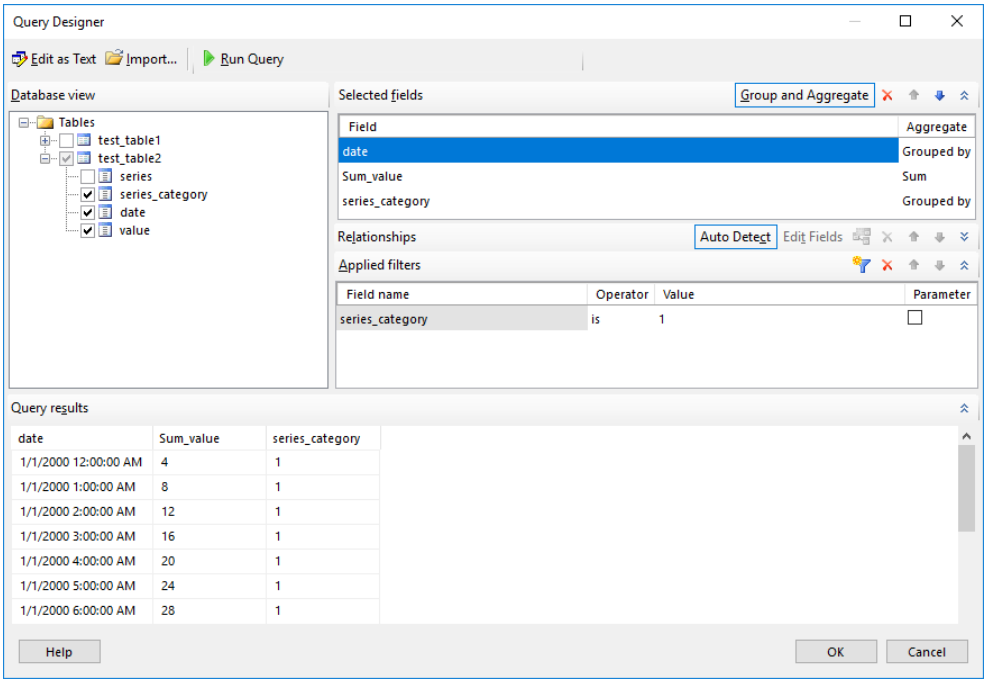
4.1.15 Příklad na datové operace 2 – Query Designer

V této podkapitole zkusím připravit data již v Query Designeru, což je zajímavé pro návrh GUI, nicméně pro realizaci příkladu v nástroji SSRS nakonec bude zvolen jiný postup, který je v následující podkapitole.

V Report Builderu tedy opět začnu tvorbou zdroje dat na testovací databázi a pokračuji tvorbou datasetu nad tímto zdrojem, resp. jeho popsání SQL dotazem v Query Designeru.

Výběr jedné časové řady se daří dle úvahy o dotazu. V Query Designeru jsem vybral všechny sloupce a dále jsem nastavil filtr popsaný podmínkou, že hodnota ve sloupci s názvem časové řady je roven „two“ (viz obr. 4.28). Filtr je přeložen do klauzule WHERE v SQL dotazu.

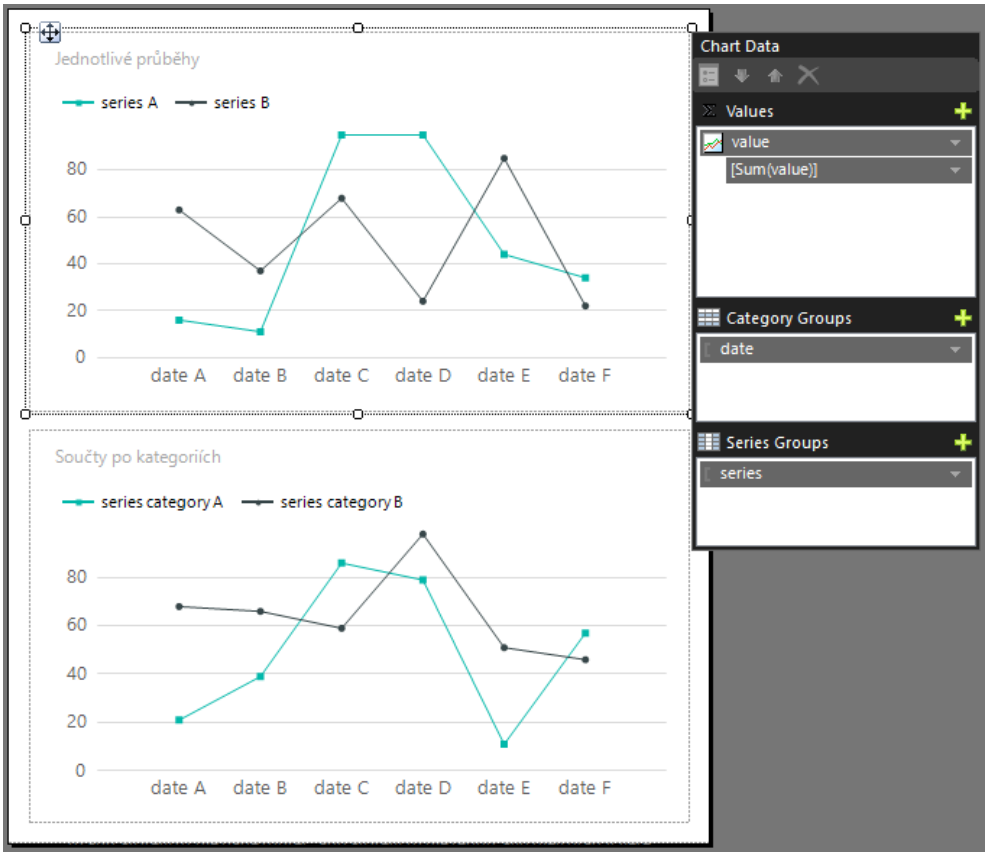
Výběr součtu časových řad kategorie 1 je také úspěšný pomocí Query Designeru. Nejprve jsem zvolil sloupce datum, hodnota a kategorie. Poté jsem kategorii pomocí filtru omezil na kategorii 1. Dále jsem zapnul funkci „Group and Aggregate“ (viz obr. 4.29), která zřejmě indikuje použití klauzule GROUP BY. U jednotlivých sloupců jsem musel nastavit, jakou hrají v seskupení roli – podle sloupce data a kategorie (která je ale vybraná jen jedna) jsou hodnoty seskupeny a na sloupec hodnot je proveden součet.



Obrázek 4.29: Query Designer – výběr součtu kategorie v druhém příkladu

```
SELECT
  test_table2.[date]
  ,SUM(test_table2.[value]) AS Sum_value
  ,test_table2.series_category
FROM
  test_table2
WHERE
  test_table2.series_category = 1
GROUP BY
  test_table2.[date]
  ,test_table2.series_category
```

Zdá se, že musím mít pro každý výsledný sloupec vlastní dataset. Vybraný sloupec hodnot z dotazu je totiž vždy jen jeden, i když obsahuje hodnoty z více časových řad. Pokud bych chtěl mít více výsledných sloupců, potřeboval bych výsledný sloupec (obsahující více časových řad v řádcích pod sebou) nějak rozdělit do více sloupců, což ale Query Designer neumožňuje.



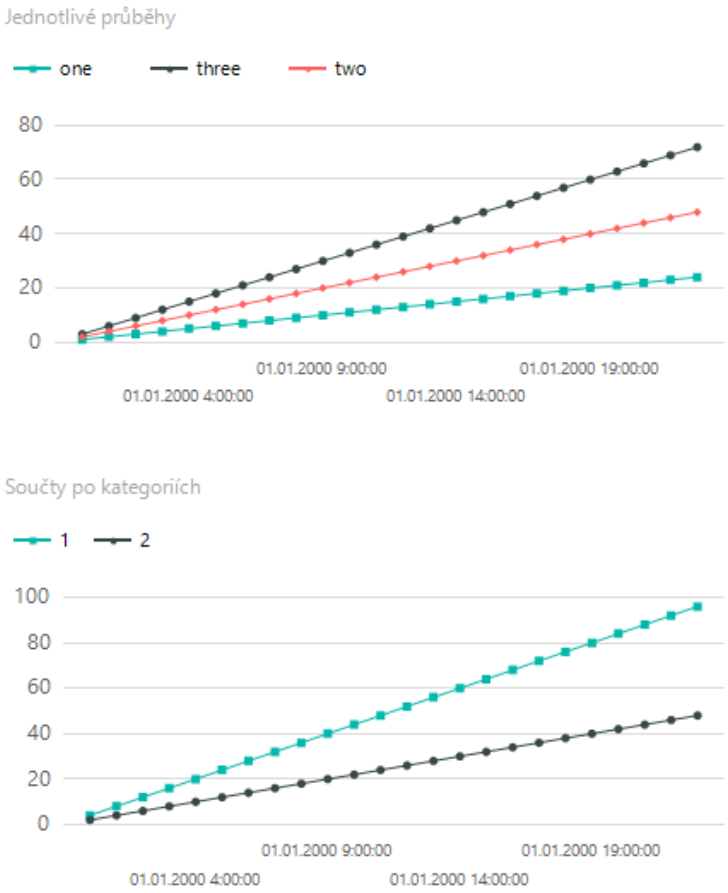
Obrázek 4.30: Grafy druhého příkladu v režimu Designer

4.1.16 Příklad na datové operace 2 – napojení na grafy a tabulky

Při napojení dat na graf a tabulku narážím na problém – zobrazovací komponenta může mít pouze jeden zdrojový data set – což dává smysl v kontextu s nalezeným elementem DataSetName v RDL (viz rozbor RDL u předchozího příkladu). Jediný způsob jak do zobrazovací komponenty dostat všechna potřebná data jako jeden dataset je tedy použít celou tabulku, neboť potřebuji použít všechna její data. Jak již bylo naznačeno v předchozí podkapitole, tak tedy dotaz vytvořený v Query Designer nebude použit.

Před zobrazením dat v zobrazovací komponentě na data lze aplikovat výraz, ale při procházení jeho možností zjišťuji, že data seskupovat neumí. Všiml jsem si ale, že zobrazovací komponenty mají interní možnost seskupování dat, což by mohlo být, to co potřebuji.

U grafů se mi provedení tohoto seskupení daří (viz obr. 4.30). Pro zobrazení jednotlivých průběhů vyberu v nastavení grafu (pravá část obrazovky) do parametru Values sloupec s hodnotami, do parametru Category Groups

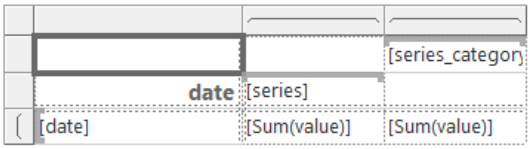


Obrázek 4.31: Grafy druhého příkladu v režimu Run

sloupec s daty a do parametru Series Groups sloupec s názvem časové řady. Pro zobrazení součtů po kategoriích volím Values a Category Groups stejným způsobem, pouze do Series Groups vyberu sloupec kategorie – tedy aby proběhlo seskupení po kategoriích.

Data jsou tedy seskupena v grafech, jen jsem nenašel způsob jak časové řady z obou grafů (viz obr. 4.31) sloučit do jednoho, jak je to u schématu k tomuto příkladu s datovými pohledy (viz obr. 4.26). Zřejmě to ani nelze, protože graf zobrazuje data jen z jednoho datasetu a pouze zvolí jak má tato data zobrazit. Pro sloučení bych ale potřeboval data zobrazit dvakrát v jednom grafu – jednou po jednotlivých řadách a jednou po kategoriích. Na toto sloučení bych v tvorbě datasetu, tedy v Query Designeru, potřeboval využít funkce UNION, podobně jako v úvaze o dotazu (viz podkapitola 4.1.14) a sloučit neagregovanou tabulku samu se sebou v agregované variantě. Tuto možnost však v Query Designeru nenacházím.

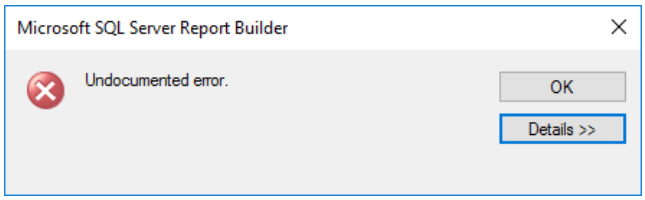
Na druhou stranu jsem nakonec ani nemusel využít zjednodušení, které



Obrázek 4.32: Tabulky druhého příkladu v režimu Designer

				1	2
date	one	three	two		
01.01.2000 0:00:00	1	3	2	4	2
01.01.2000 1:00:00	2	6	4	8	4
01.01.2000 2:00:00	3	9	6	12	6
01.01.2000 3:00:00	4	12	8	16	8
01.01.2000 4:00:00	5	15	10	20	10
01.01.2000 5:00:00	6	18	12	24	12
01.01.2000 6:00:00	7	21	14	28	14
01.01.2000 7:00:00	8	24	16	32	16
01.01.2000 8:00:00	9	27	18	36	18

Obrázek 4.33: Tabulky druhého příkladu v režimu Run



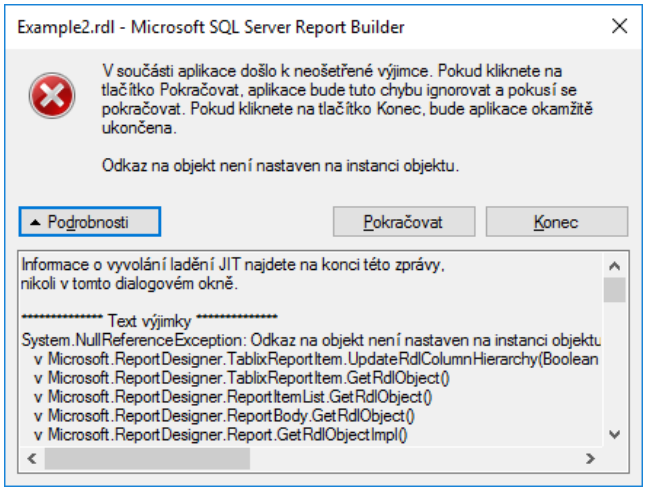
Obrázek 4.34: Neošetřená chyba při nastavení tabulky 1

jsem uvažoval při tvorbě SQL dotazu, tedy, že vím dopředu počet a názvy kategorií. Seskupení v grafu provedlo rozdělení do skupin bez této znalosti, jako v obecnější variantě uvažovaného SQL dotazu.

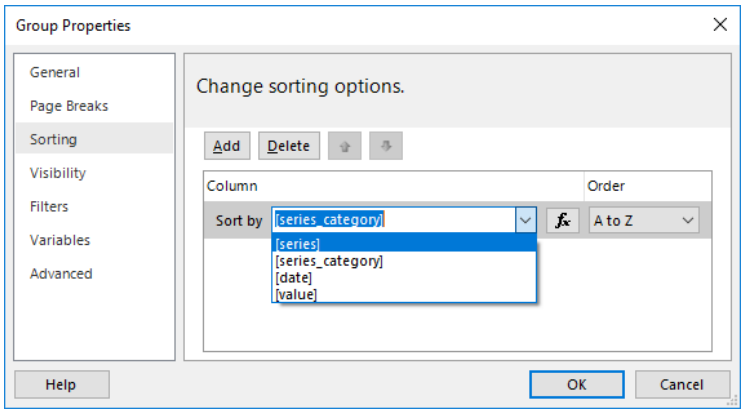
Vyzkouším totéž i u tabulek. Daří se mi dostat podobným způsobem jako v grafu dokonce všechny sloupce (agregované i neagregované časové řady) do jedné tabulky (na rozdíl od grafu), neboť v Designeru je možné předepsat seskupení pro skupiny buněk tabulky, kterých ale může být v tabulce více (viz obr. 4.32 a 4.33).

Mám zde ale jiný problém, nedaří se mi ale popisky těchto skupin umístit na stejný řádek. Při různých pokusech o sjednocení popisků na stejný řádek nástroj vyhazuje různé neošetřené chyby (viz obr. 4.34 a 4.35).

Sloupce se součtem po kategoriích jsou pojmenovány podle hodnoty kate-



Obrázek 4.35: Neošetřená chyba při nastavení tabulky 2



Obrázek 4.36: Nastavení řazení v SSRS

gorie (viz obr. 4.33), což není smysluplný název – chtěl bych, aby se jmenovaly jako „Součet kategorie XY“.

Hledám tedy mechanismus podobný předpisu názvu a použitím tzv. placeholderů, který umožňují datové pohledy pro agregace se seskupením v našem Jazyce. Díky nim je v datových pohledech se seskupením možné definovat jak se mají výsledné sloupce nazývat.

Při hledání náhodou nalézám i možnost agregované sloupce řadit (viz obr. 4.36), což je to, co dělá v původním příkladu pohled pro řazení. Je zde možné vybrat, dle čeho bude řazení provedeno. Dále mohu nechat sloupce seřadit vzestupně nebo sestupně.

Mechanismus s placeholdery sice nenacházím, ale zjišťuji, že předpis názvů sloupců je dán výrazem. Tento výraz se mi daří upravit na požado-

date	one	three	two	Součet kategorie 2	Součet kategorie 1
01.01.2000 0:00:00	1	3	2	2	4
01.01.2000 1:00:00	2	6	4	4	8
01.01.2000 2:00:00	3	9	6	6	12
01.01.2000 3:00:00	4	12	8	8	16
01.01.2000 4:00:00	5	15	10	10	20
01.01.2000 5:00:00	6	18	12	12	24
01.01.2000 6:00:00	7	21	14	14	28
01.01.2000 7:00:00	8	24	16	16	32
01.01.2000 8:00:00	9	27	18	18	36

Obrázek 4.37: Tabulky druhého příkladu s pojmenovanými sloupci

```
1    <?xml version="1.0" encoding="utf-8"?>
2    <Report MustUnderstand="df"
3            xmlns="http://schemas.microsoft.com/sqlserver/reporting/2016/01/reportdefinition"
4            xmlns:rd="http://schemas.microsoft.com/SQLServer/reporting/reportdesigner"
5            xmlns:df="http://schemas.microsoft.com/sqlserver/reporting/2016/01/reportdefinition/defaultfontfamily">
6      <df:DefaultFontFamily>Segoe UI</df:DefaultFontFamily>
7      <AutoRefresh>0</AutoRefresh>
8      <DataSources>...</DataSources>
19     <DataSets>...</DataSets>
61     <ReportSections>...</ReportSections>
1289   <ReportParametersLayout>...</ReportParametersLayout>
1295   <rd:ReportUnitType>Inch</rd:ReportUnitType>
1296   <rd:ReportID>6fc409d6-40c6-44a9-b8a9-fb3f9c60afb0</rd:ReportID>
1297 </Report>
```

Obrázek 4.38: RDL předpis druhého příkladu

vaný název sloupců pomocí trochu neintuitivní (programátorské) úpravy z původního „=Fields!series_category.Value“ na „="Součet kategorie " + Fields!series_category.Value.ToString()“. Výsledná tabulka s touto úpravou a se změnou seřazení kategorií je vidět na obr. 4.37.

4.1.17 Příklad na datové operace 2 – náhled do definice reportu

Následuje náhled do definice vytvořeného reportu v RDL (viz obr. 4.38). Celý report ještě rozsáhlejší než u předchozího příkladu, má celkem 1294 řádků, zaměřím se z něj opět pouze na zajímavé věci v kontextu operací s daty. Celý RDL dokument je k nalezení na přiloženém DVD.

Element se zdroji dat a element s datasety (viz obr. 4.39) jsou principem stejné jako v příkladu 1.

V elementu ReportSections (viz obr. 4.40, cca 850 řádek – většina ale popis stylů) očekávaně nacházím definici tabulky a dvou grafů.

```
<DataSources>
  <DataSource Name="DataSource1">
    <ConnectionProperties>
      <DataProvider>SQL</DataProvider>
      <ConnectString>Data Source=localhost\SQLEXPRESS;Initial Catalog=test_db</ConnectString>
      <IntegratedSecurity>true</IntegratedSecurity>
    </ConnectionProperties>
    <rd:SecurityType>Integrated</rd:SecurityType>
    <rd:DataSourceID>09cb44c3-9fd8-4974-98a1-794f67606ab5</rd:DataSourceID>
  </DataSource>
</DataSources>

<DataSets>
  <DataSet Name="DataSet1">
    <Query>
      <DataSourceName>DataSource1</DataSourceName>
      <CommandText>SELECT
  test_table2.series
  ,test_table2.series_category
  ,test_table2.[date]
  ,test_table2.[value]
FROM
  test_table2</CommandText>
      <rd:DesignerState>...</rd:DesignerState>
    </Query>
    <Fields>...</Fields>
  </DataSet>
</DataSets>
```

Obrázek 4.39: Sekce DataSources a DataSets v RDL druhého příkladu

```
<ReportSections>
  <ReportSection>
    <Body>
      <ReportItems>
        <Tablix Name="Tablix3">...</Tablix>
        <Chart Name="Chart2">...</Chart>
        <Chart Name="Chart3">...</Chart>
      </ReportItems>
      <Height>6.875in</Height>
      <Style>...</Style>
    </Body>
    <Width>5.04042in</Width>
    <Page>...</Page>
  </ReportSection>
</ReportSections>
```

Obrázek 4.40: Sekce ReportSections v RDL druhého příkladu

```
540                 <Group Name="Chart2_SeriesGroup">
541                   <GroupExpressions>
542                     <GroupExpression>=Fields!series.Value</GroupExpression>
543                   </GroupExpressions>
544                 </Group>
```

Obrázek 4.41: Sekce Group v RDL druhého příkladu

Nastavení parametrů grafů pro napojení na data (viz obr. 4.41) se oproti prvnímu příkladu se liší tím, že je zde výrazem nastaveno seskupení po časových řadách – *<GroupExpression>=Fields!series.Value</GroupExpression>* u prvního grafu a u druhého grafu obdobně seskupení po kategoriích – *<GroupExpression>=Fields!series_category.Value</GroupExpression>*.

Tabulka je definována podobně jako graf, jen je zde definice šablon jednotlivých buněk tabulky, na které jsou mapovány sloupce z výsledku SQL dotazu. Po spuštění jsou pak šablony buněk nahrazeny celými sloupci dat. Seskupování je provedeno stejně jako v grafu, tedy elementem GroupExpression.

4.1.18 Příklad na datové operace 2 – shrnutí

Druhý příklad byl komplikován složitějším mapováním časových řad na relační databázi. Je otázkou, jestli by nebyl vhodnější jiný typ databáze, který by umožňoval přímočařejší práci s daty povahy časových řad.

Po odsunutí operací s daty z SQL dotazu, kde to nebylo příliš použitelné (dataset obsahoval pouze jeden sloupec a zobrazovací komponenta tím pádem také), do nastavení zobrazovacích komponent, bylo nakonec přijatelně obtížné pokročilé operace s daty provést.

Nepodařilo se mi sice vyladit drobnosti jako sloučení obou grafů (s jednotlivými a s agregovanými průběhy) do jednoho a umístění popisků tabulky na stejný řádek, ale kromě toho, se zdařilo nasimulovat tyto složitější operace prováděné datovými pohledy pomocí testovaného nástroje SSRS.

Shrnutí poznatků z jednotlivých zkoumaných nástrojů se nachází v závěru celé kapitoly, dále následuje průzkum dalších nástrojů.

4.2 Nástroj 2 – Crystal Reports (neúspěch)

Dalším zkoumaným nástrojem je SAP Crystal Reports. Nástroj se stejně jako SSRS zabývá tvorbou reportů z různých typů zdrojů dat. Crystal Reports nabízí také širokou škálu zobrazovacích komponent, podobně jako SSRS, včetně spojnicových grafů a tabulek, které jsou relevantní pro tuto práci. Existuje v něm také možnost exportovat výsledné reporty do PDF nebo do formátů nástrojů z balíčku Microsoft Office [19].

Nástroj je typu trialware, je tedy dostupná verze k vyzkoušení po dobu 30 dní zdarma, kterou se pokusím zprovoznit. Před stažením ze stránek výrobce

je nutné se registrovat, čímž ke stažení získávám instalátor a kód k trialware licenci.

Při instalaci však narážím na problém – instalátor hlásí, že zadaný obdržený kód je neplatný. Výrobce radí obrátit se na jeho oficiální fórum, kde nacházím stejný problém a k němu několik postupů k vyřešení. Po vyzkoušení postupů se mi však stále nedaří chybu instalátoru vyřešit. Dále jsem zkoušel instalaci na jiném počítači s jinou verzí OS Windows (Windows 7 a Windows 10) a pokusil jsem se software znovu registrovat a stáhnout, ale se stejným výsledkem, tedy bez úspěchu.

I přes neúspěšnou instalaci, se z prozkoumaných zdrojů zdá, že je nástroj Crystal Reports v mnoha ohledech podobný nástroji SSRS [19]. Doufám tedy, že nemožnost jeho vyzkoušení nebude mít zásadní vliv na potřebné výstupy z průzkumu a zaměřuji se tedy na další nástroje.

4.3 Nástroj 3 – Syncfusion Dashboard

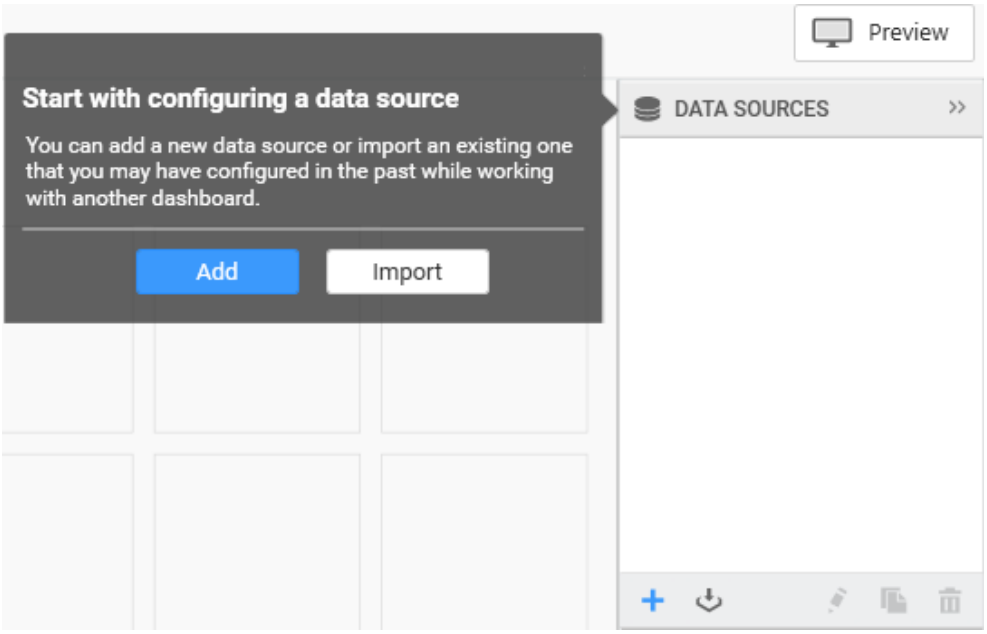
Třetím zkoumaným nástrojem je Syncfusion Dashboard, který se zabývá tvorbou tzv. dashboardů.

Dashboard je podobný koncept jako report, oboje je výstup dat v pro člověka čitelné podobě. Report je typicky rozložen do jednotlivých stránek a je tak vhodnější pro tisk, na druhou stranu dashboard je typicky přístupný přes webové rozhraní. Report bývá spíše statický, zatímco dashboard má zpravidla více interaktivních prvků [20]. Grafické prvky v reportech v této práci označuji jako zobrazovací komponenty a v dashboardu se nazývají widgety. Tyto rysy ale nejsou pevně vyhraněny a existují pro ně výjimky. Pro účely této práce není potřeba mezi těmito dvěma pojmy rozlišovat.

Pro první seznámení se softwarem jsem zhlédl některé doporučené video tutoriály od společnosti Syncfusion [21][22], na kterých probíhá prezentace a ukázka práce s nástrojem (provází jimi Daniel Jebaraj, Vice President Syncfusion).

Nástroj obsahuje výrazy (expressions) podobně jako v SSRS a pomocí nich je možné definovat sloupce vzniklé provedením operací nad zdrojovými sloupci (podobně jako v našem Jazyce datový pohled pro agregace). Nástroj nabízí opět širokou škálu zobrazovacích komponent – tedy widgetů. Pro účely této práce jsou však opět zajímavé pouze tabulka a spojnicový graf. Nástroj má pro editaci dashboardu režim Designer a pro zobrazení výstupu režim Preview (opět podobně jako u SSRS) [21].

Vývoj dashboardů lze rozdělit do tří fází – příprava dat, tvorba dashboardu a jeho zveřejnění. To je opět možné přirovnat k SSRS, kde je příprava dat přípravou datových zdrojů a datasetů, tvorba dashboardu je v SSRS zastoupena tvorbou reportu a nasazení je v SSRS spuštění, či export výsledného reportu. U Syncfusion Dashboard fáze zveřejnění znamená nahrání dashboardu na Dashboard server, odkud je dostupný pro ostatní uživatele prostřed-



Obrázek 4.42: Syncfusion – ukázka zabudované nápovědy

nictvím webového prohlížeče. Je zde také možné omezit přístup k dashboardu nastavením autentizace uživatelů [22].

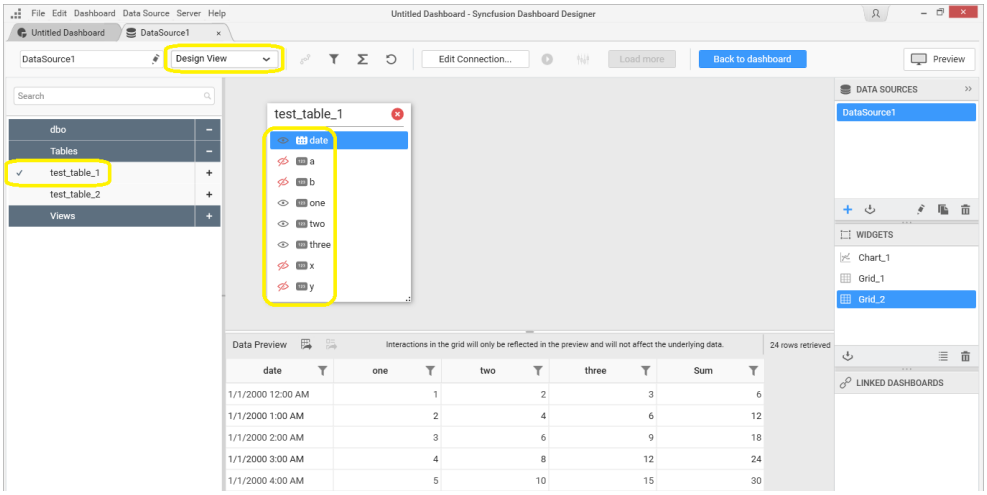
Stejně jako u Crystal Reports je Syncfusion Dashboard trialware s dostupnou 30 denní zkušební verzí. Před stažením je nutné se registrovat. Instalace proběhla bez problémů a dále tedy budu pracovat s nástrojem pro tvorbu dashboardů – Dashboard Designer (verze 2.1.0.2).

4.3.1 Příklad na datové operace 1

Následující příklad se zabývá vyzkoušením některé funkcionality datových pohledů v tomto nástroji. Jedná se o stejný příklad jako u prvního zkoumaného nástroje SSRS – viz podkapitola 4.1.9. Ve zmíněné podkapitole se nachází krátký popis a vysvětlení příkladu a před čtením dále je vhodné si jej přečíst. Tento a následující příklad jsou realizovány ve všech zkoumaných nástrojích, jak již bylo zmíněno v úvodu celé kapitoly 4.

Při prvním používání nástroje si všímám moderního vzhledu a zabudované nápovědy a tipů na intuitivních místech v aplikaci (viz obr. 4.42, což u nástroje SSRS nebylo).

Prvním krokem je napojení k mé testovací sadě dat. Napojení dat je zde stejně jako u SSRS prostřednictvím zdrojů dat (data sources). Tyto zdroje mohou být napojeny na různé druhy relačních databází a dále také například na dokument Microsoft Office Excel nebo na CSV soubor. Má testovací sada dat je v Microsoft SQL databázi, použiju tedy tento typ. Pro připojení vy-



Obrázek 4.43: Syncfusion – tvorba zdroje dat prvního příkladu

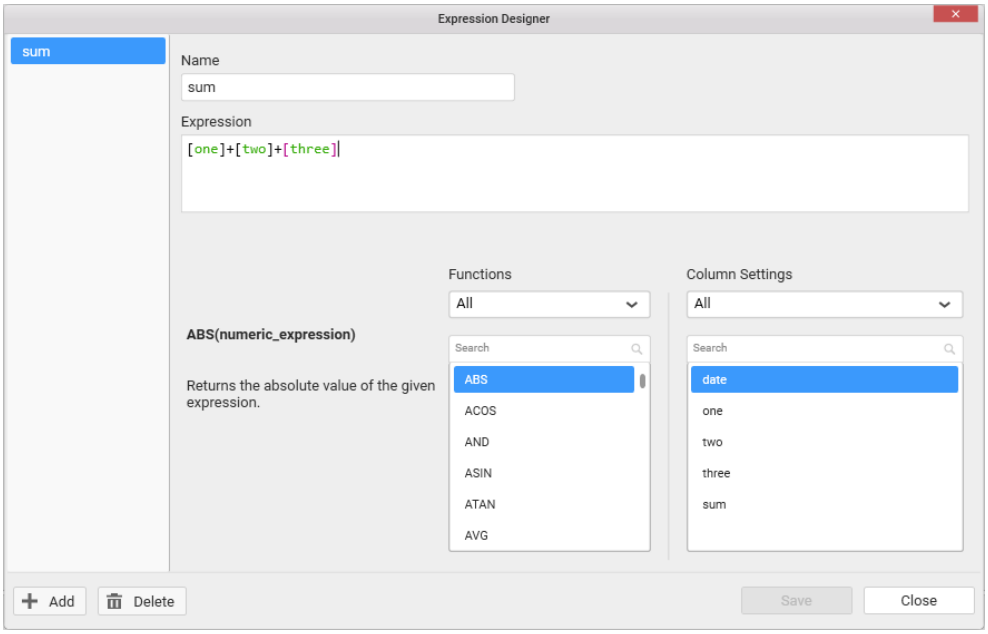
bírám můj testovací server a na něm běžící databázi – tento výběr proběhl v samostatném formuláři.

Dalším krokem je vybrat tabulky a sloupce, které budu potřebovat – pro to je v nástroji designer (viz obr. 4.43). Vybírám testovací tabulku (zaškrtnutí test_table_1) a v této tabulce je možné schovat sloupce, schovám tedy sloupce a, b, x a y (zaškrtnuté ikonky oka).

Dále přidávám sloupec pro součet zbývajících sloupců (viz obr. 4.44). Tento sloupec je popsán výrazem. Pro editaci výrazu je zde Expression Designer, který našeptává při psaní a dále jsou zde pomocné seznamy s dostupnými sloupci a dalšími funkcemi. Tvořím tedy výsledný výraz pro součet „[one]+[two]+[three]“.

V designeru zdroje dat lze přepnout i do režimu, kde je náhled SQL dotaz (Code View, viz obr. 4.43 – roletka s vybraným Design View). Stejně jako u SSRS ale není možné provést manuální úpravu v dotazu a následně se vrátit do designeru. Vygenerovaný SQL dotaz, obsahuje i součtový sloupec, který vznikl pomocí sčítání sloupců v klauzuli SELECT, jak jsem předpokládal v rozboru SQL dotazu (viz podkapitola 4.1.9 u SSRS). Podoba vygenerovaného dotazu je následující:

```
SELECT top(100)
[dbo_test_table_1].[date] AS [dbo_test_table_1_date]
,[dbo_test_table_1].[one] AS [dbo_test_table_1_one]
,[dbo_test_table_1].[two] AS [dbo_test_table_1_two]
,[dbo_test_table_1].[three] AS [dbo_test_table_1_three]
,[dbo_test_table_1].[one] + [dbo_test_table_1].[two] +
  [dbo_test_table_1].[three] AS [Sum]
FROM [dbo].[test_table_1] AS [dbo_test_table_1]
```



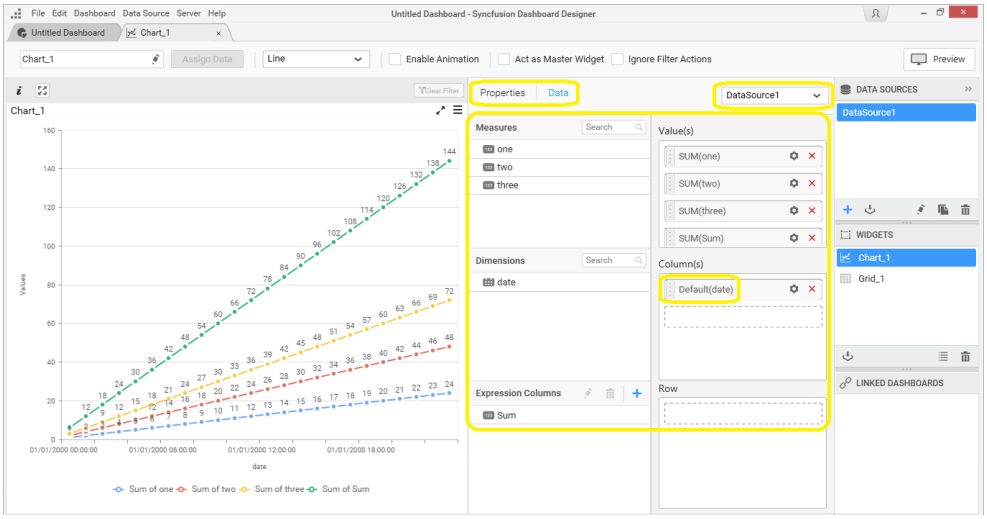
Obrázek 4.44: Syncfusion – tvorba součtového sloupce pomocí výrazu

Dále chci přidat tabulku a spojnicový graf zobrazující tato data. Začnu grafem – z dostupné škály widgetů tedy vybírám spojnicový graf (*Line Chart*). Graf nejprve přidám na dashboard a dále se zabývám jeho nastavením. Nastavení grafu je na samostatné obrazovce, kde na levé části je náhled grafu a na pravé části je nastavení grafu rozděleno do sekcí *Properties* a *Data* (viz obr. 4.45). Pro napojení dat přecházím do sekce *Data*, v ní vyberu můj připravený zdroj dat a dále musím vybrat, co z tohoto zdroje chci v grafu zobrazit.

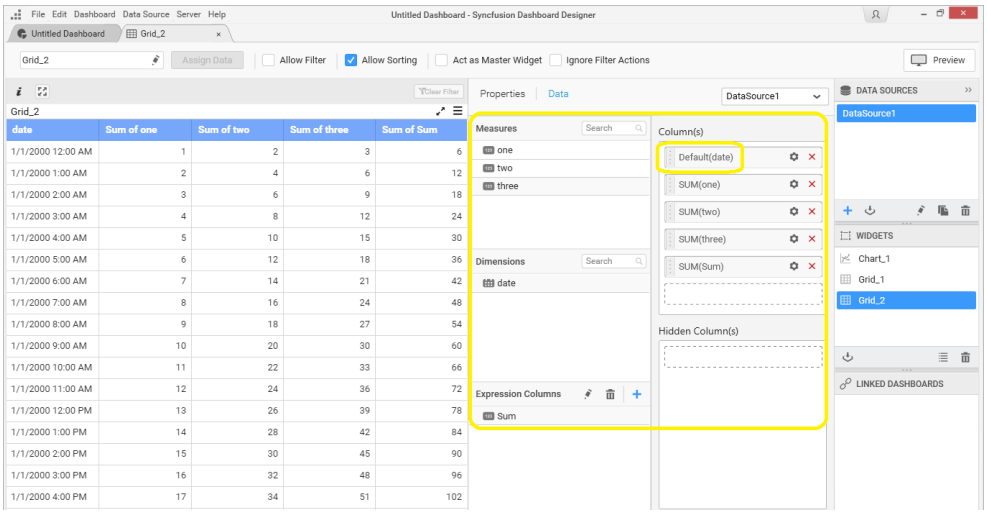
Dostupné sloupce ze zdroje dat jsou rozděleny do tří kategorií:

1. *Measures* – zde se nachází sloupce *one*, *two* a *three*,
2. *Dimensions* – zde se nachází sloupec *date*,
3. *Expression Columns* – zde se nachází sloupec se součtem – *Sum*.

Tyto dostupné sloupce mohu přiřadit do tří kategorií dat grafu – *Values*, *Columns* a *Row*. Toto rozdělení mi nepřijde příliš intuitivní, nicméně po delším zkoušení zjišťuji, že do kategorie *Columns* musím dát sloupec datum (tato kategorie tedy zřejmě určuje hodnoty na ose X grafu), je ale nutné přepnout projekční funkci zobrazující datum z funkce *Year* na funkci *Hours* nebo lépe na funkci *Default*. Původně vybraná funkce *Year* totiž vzhledem k tomu, že jsou data pouze pro jeden den, promítnula 24 hodnot ve sloupci do jedné hodnoty s číslem roku. Do kategorie *Row* vyberu sloupce *one*, *two*, *three* a dále sloupec se součtem a graf funguje, tak jak potřebuji (viz obr. 4.45). Tedy kromě toho,



Obrázek 4.45: Syncfusion – napojení grafu na data v prvním příkladu

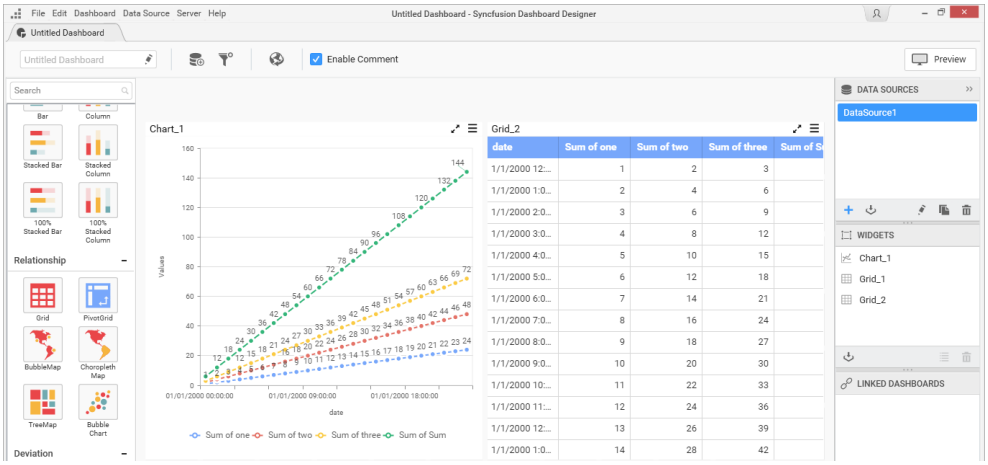


Obrázek 4.46: Syncfusion – napojení tabulky na data v prvním příkladu

že byla časovým řadám automaticky vygenerována do názvu předpona „Sum of“, toto pojmenování lze však libovolně změnit v sekci *Properties*.

Dále přidám tabulku, tedy widget pojmenovaný jako *Grid*. Opět přejdu do sekce nastavení dat, kde jsou ve stejných kategoriích jako u grafu, dostupné sloupce ze zdroje dat (viz obr. 4.46). Kategorie, do kterých mám sloupce přiřadit, se ale liší, jsou zde pouze dvě – *Columns* a *Hidden Columns*.

Všechny sloupce umisťuji do kategorie *Columns*, jen je nutné opět změnit funkci pro zobrazení data z *Year* na *Default*, aby z tabulky nevznikl jen jeden řádek. Tím mám nastaven widget tabulky.



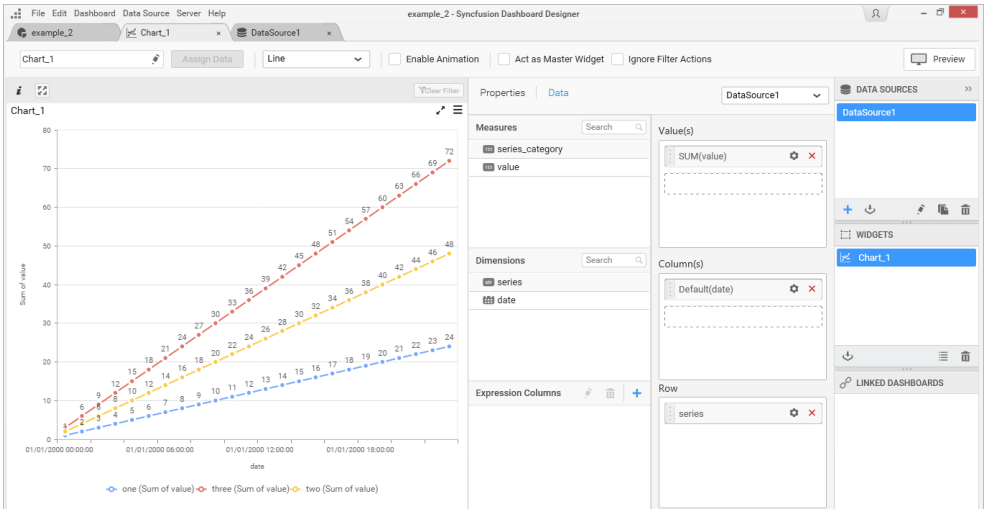
Obrázek 4.47: Syncfusion – hotový první příklad při editaci

Obrázek 4.48: Syncfusion – hotový první příklad pri spuštění

V designeru dashboardu pak vidím navržený dashboard a pomocí tlačítka *Preview* se mi otvírá i skutečná podoba ve webovém prohlížeči (viz obr. 4.47 a 4.48). Zajímavé je, že v designeru jsou vidět už skutečná data, nikoli pouze ukázková, jako tomu bylo v SSRS (Full Name 1, Full Name 2 atd.). Podle mého názoru by z principu i v SSRS mohla být v designeru vidět skutečná data pro lepší představu.

Rozmístění widgetů v dashboardu lze měnit a widgety lze roztahovat ale pouze po předem připravené mřížce, což usnadňuje zarovnání widgetů, nicméně může být i omezující.

Celý dashboard je uložen jako „.sydx“ soubor. Tento soubor je binární povahy a nepodařilo se mi o něm najít bližší podrobnosti. Zdá se však, že definice dashboardu není přístupná v nějakém doménově specifickém jazyce, na rozdíl od SSRS, kde je jazyk RDL. Účel průzkumu je však hlavně inspirace



Obrázek 4.49: Syncfusion – napojení grafu 1 na data v druhém příkladu

z GUI daných nástrojů, takže tento fakt nesnižuje význam dalšího zkoumání nástroje Syncfusion Dashboard.

Dále je možnost zobrazený widget ve webovém prohlížeči exportovat do souborů formátů jako PDF, jako obrázek nebo jako soubor Microsoft Office Excel. Exportované dashboardy tohoto i následujícího druhého příkladu jsou ve formátu PDF uloženy na přiloženém DVD.

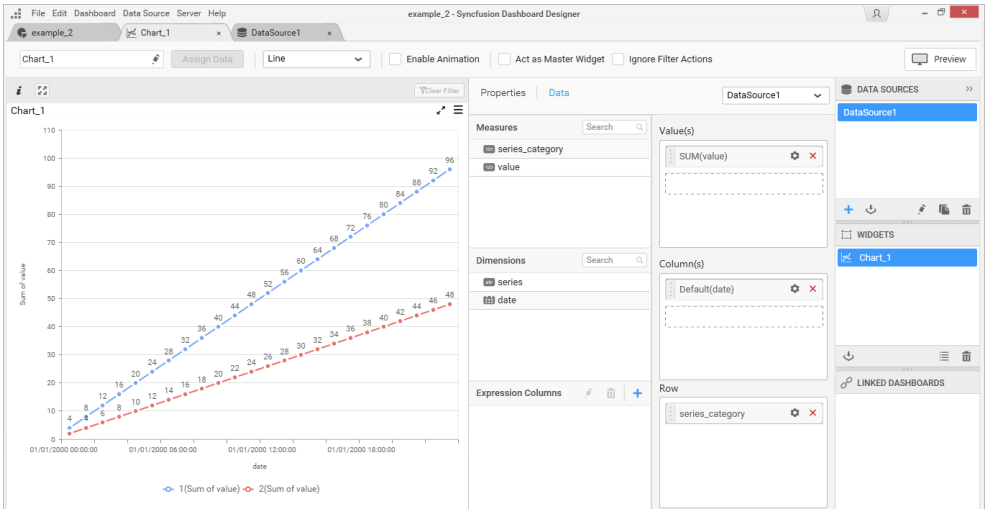
4.3.2 Příklad na datové operace 2

Druhý příklad podobně jako první testuje funkcionalitu našeho Jazyka a datových pohledů v tomto nástroji. Jedná se o příklad použitý při testování nástroje SSRS a jeho bližší popis se nachází v podkapitole 4.1.13 u průzkumu SSRS.

Začnu tedy tvorbou zdroje dat. Připojím jej ke své testovací databázi a vyberu tabulku s testovacími daty k tomuto příkladu, kde ponechám všechny sloupce. Nenacházím zde možnost seskupení sloupců (podobně jako klauzule GROUP BY v SQL), které potřebuji provést a tak nechávám sloupce neseskupené a seskupení a řazení realizuji až při tvorbě widgetů.

Nejprve přidám graf. Při nastavení dat se mi však hned nedaří provést seskupení, jak bych potřeboval, a až po delším experimentování se mi podařilo dostat do grafu časové řady pro každou časovou řadu jednu s hodnotami na ose Y a s datem na ose X – přiřazením sloupce *values* do kategorie *Values*, sloupce *date* do kategorie *Columns* a sloupce *series* do kategorie *Row* (viz obr. 4.49).

Podobně je pak možné zobrazit součty jednotlivých skupin kategorií, když změním obsah sekce *Row* ze sloupce *series* na sloupec *series_category* (viz



Obrázek 4.50: Syncfusion – napojení grafu 2 na data v druhém příkladu

obr. 4.50). Nedaří se mi však do jednoho grafu dostat jak jednotlivé časové řady, tak jejich součty po kategoriích (do kategorie *Row* lze přiřadit nejvýše jeden sloupec). Tento problém jsem měl i v SSRS. Na dashboard tedy umisťuji dva grafy.

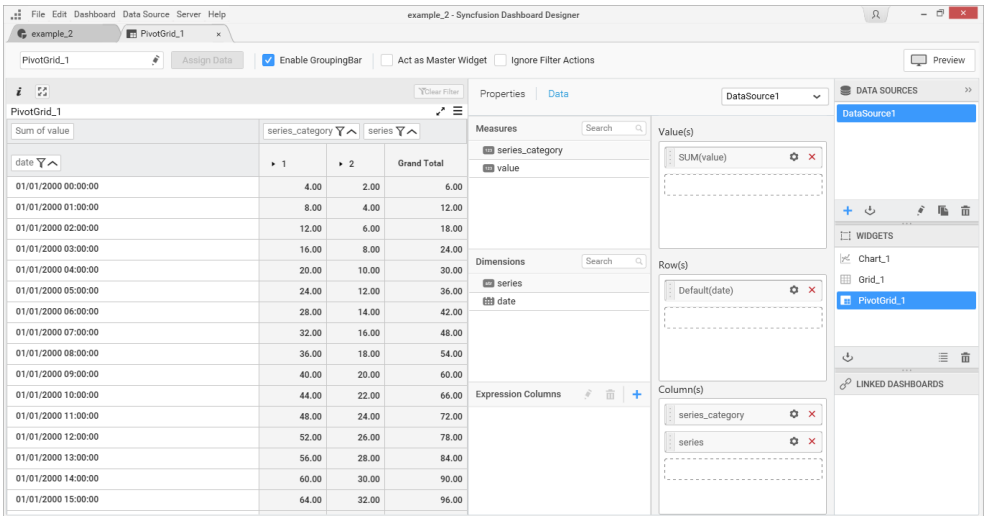
Nakonec tedy zkusím přidat tabulku, ve které bych opět chtěl zobrazit jednotlivé časové řady a zároveň jejich součty po kategoriích. Widget Grid použitý v prvním příkladě však neumí seskupovat data vůbec, pomocí widgetu Pivot Grid však toto možné je a na rozdíl od grafu, zde lze zobrazit pohromadě jednotlivé časové řady i s jejich součty.

Pivot Grid má v nastavení dat podobné kategorie jako graf, jen místo kategorie Row je zde kategorie Rows, tedy může obsahovat více sloupců a právě díky tomuto je možné v tabulce zobrazit jak jednotlivé časové řady (tedy sloupec hodnot seskupit podle sloupce názvů časových řad), tak součty časových řad po kategoriích (tedy sloupec hodnot seskupit podle sloupce kategorií časových řad). Při odpovídajícím nastavení tedy získávám tabulku s daty požadovanou v zadání (viz obr. 4.51).

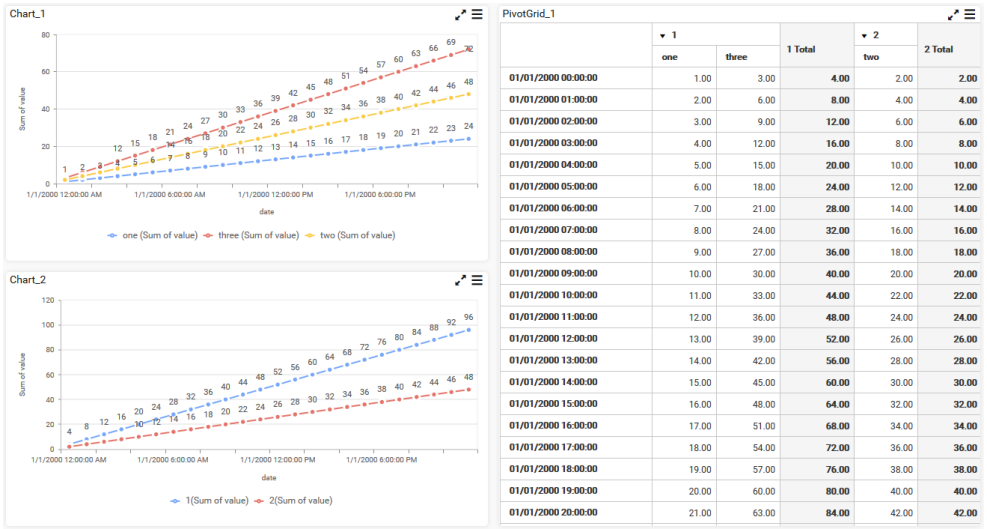
Sloupce v tabulce se zároveň automaticky umístily do skupin, které lze v dashboardu rozbalovat a sbalovat, což je funkcionalita, která se v našem Jazyce u datagridu také řeší.

Výsledná podoba dashboardu se nachází na obrázku 4.52.

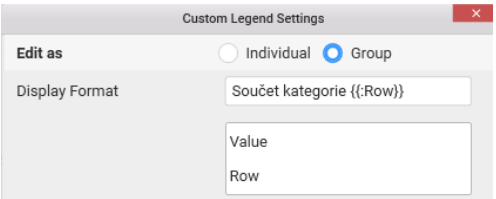
Jak je vidět na obrázku 4.52, nástroj vygeneroval pro sloupce v tabulce (resp. časové řady v grafu), vzniklé součtem časových řad dané kategorie, názvy. Tyto názvy jsou pouze hodnota dané kategorie nebo v grafu pak hodnota dané kategorie s popiskem „Sum of value“. Podobně jako v našem jazyce a také při řešení tohoto příkladu v SSRS chci sloupcům určit název dle předpisu „Součet kategorie XY“.



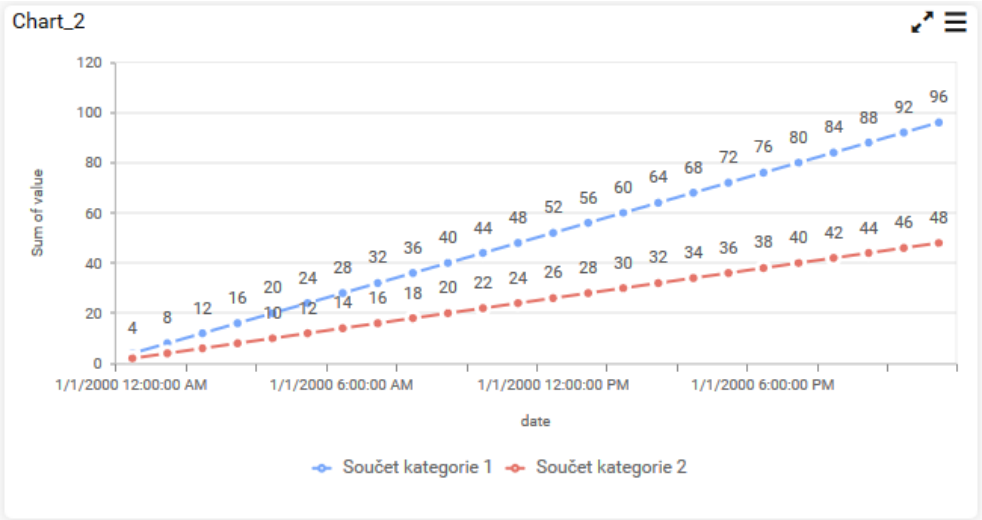
Obrázek 4.51: Syncfusion – napojení tabulky na data v druhém příkladu



Obrázek 4.52: Syncfusion – vysledný dashboard druhého příkladu



Obrázek 4.53: Syncfusion – předpis názvu agregovaných řad v grafu



Obrázek 4.54: Syncfusion – graf s pojmenovanými agregovanými řadami

U grafu se mi to daří – konkrétně v nastavení legendy grafu (viz obr. 4.53). V tomto nastavení se nachází *Display Format*, což je řetězec určující název výsledným sloupcům z agregace. V tomto řetězci je možné používat tzv. placeholder („{{:Row}}“, podobně jako v datovém pohledu pro agregace se seskupením v našem Jazyce), za který se v konečném zobrazení dosadí hodnota sloupce dané skupiny, která je agregována. V případě kategorií sloupců 1 a 2, tedy hodnoty 1 a 2. Nastavením řetězce na „Součet kategorie {{:Row}}“ tak získávám v grafu požadované pojmenování (viz obr. 4.54).

Pro tabulku (widget *Pivot Grid*) jsem však tuto funkcionalitu nikde nenašel a výsledné agregované sloupce se mi u tabulky přejmenovat nepodařilo.

Nakonec jsem zkoumal ještě zmíněné seskupení do rozbalovacích skupin v tabulce, které proběhlo automaticky. Snažil jsem se toto seskupení v nastavení najít a upravit nebo ho jen vypnout. V nastavení tabulky jsem však žádný parametr, který by seskupení ovlivňoval, nenašel. Dále jsem zkoušel ručně nastavit skupiny sloupcům i pro příklad 1 v komponentě Grid, tak jak to libovolně lze v našem Jazyce v datagridu, ale ani u tohoto widgetu jsem tuto možnost nenašel. Dle mého názoru je poněkud zvláštní, pokud tabulka

nenabízí v tomto nástroji parametrizaci těchto relativně základních vlastností.

4.4 Nástroj 4 – Sisense

Dalším zkoumaným nástrojem je Sisense, určený pro tvorbu dashboardů, podobně jako předchozí nástroj Syncfusion Dashboard.

Jedná se opět o trialware, kde je ke stažení tentokrát pouze 14denní verze. Pro stažení je nutná registrace, dále se stáhne malý instalátor (pouze 2 MB) a při instalaci se teprve celý nástroj stahuje.

Po instalaci jsou k dispozici dva nástroje – Sisense ElastiCube Manager, který slouží pro definici zdroje dat a dále je zde webové rozhraní Sisense, ve kterém jsou sestavovány dashboardy.

Pro seznámení s aplikací jsem použil seznam doporučených oficiálních video tutoriálů k základům práce se software [23]. Koncepty v Sisense jsou podobné, jako u předešlých nástrojů – jsou zde zdroje dat, widgety, tedy prvky, které data na dashboard zobrazují nebo výrazy (expressions).

4.4.1 Příklad na datové operace 1

Následující příklad se zabývá vyzkoušením některé funkcionality datových pohledů v tomto nástroji. Jedná se o stejný příklad jako u již probraných nástrojů SSRS a Syncfusion Dashboard – viz jejich stejnojmenné podkapitoly 4.1.9. Ve zmíněné podkapitole u prvního nástroje se nachází krátký popis a vysvětlení příkladu.

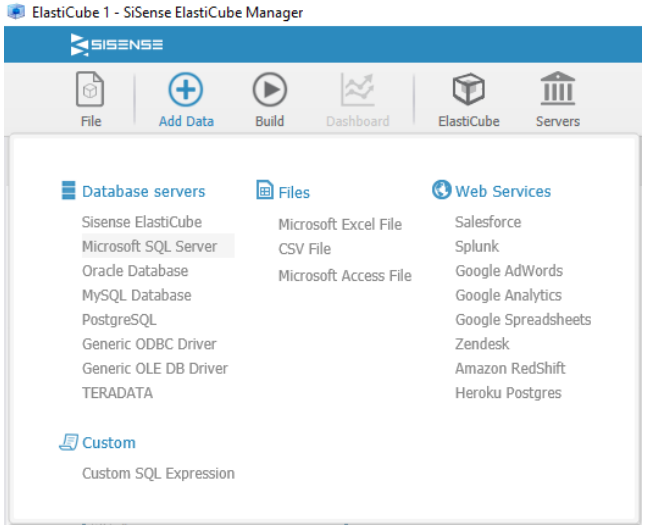
Začnu přípravou tzv. schématu, které obsahuje napojení na datový zdroj a přípravu potřebných dat z tohoto zdroje (neobsahuje tedy samotná data). Schéma je dále možné sestavit (build), čímž schéma načte skutečná data ze zdroje dat a umožní toto schéma použít při tvorbě dashboardu [24][25].

Pro tvorbu schématu použiji tedy zmíněný nástroj Sisense ElastiCube Manager, kde založím nový soubor schématu (tzv. ElastiCube soubor). Do schématu přidám připojení na datový zdroj – připravenou z SQL databázi, použitou u předchozích nástrojů. Jako zdroj dat lze použít kromě relačních databází také např. CSV nebo Microsoft Office Excel soubory nebo webové služby (viz obr. 4.55).

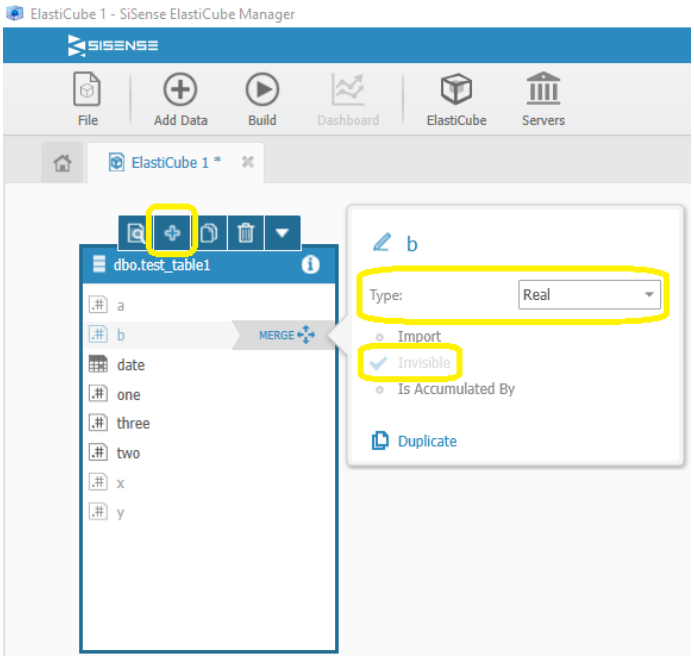
Z databáze vybírám testovací tabulku pro tento příklad a podobně jako u Syncfusion Dashboard je zde možnost skrýt nepotřebné sloupce (viz obr. 4.56).

Dále je zde možnost (podobně jako u Syncfusion Dashboard) přidat sloupec, který vznikne jako výsledek výrazu nad ostatními sloupci (tzv. Expression Column). Přidám tedy sloupec pro součet sloupců *one*, *two* a *three*:

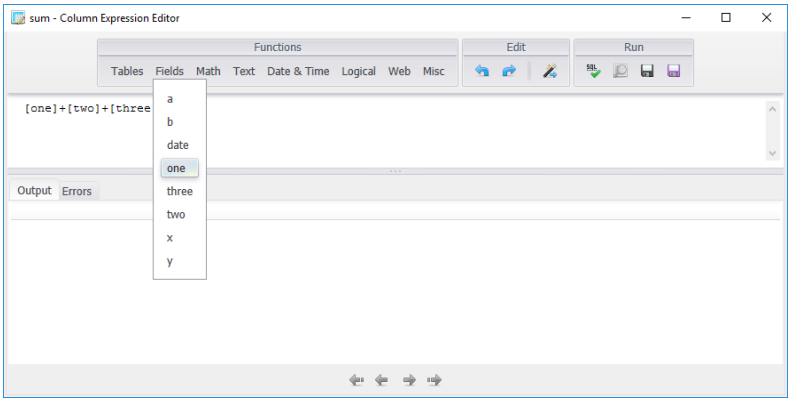
1. Pomocí tlačítka „+“ (viz obr. 4.56) přidám nový Expression Column, vyberu datový typ real (odpovídá typu u sloupců *one*, *two*, *three*) a mohu buď rovnou napsat Expression, nebo si otevřít editor na jeho tvorbu.



Obrázek 4.55: Sisense – nabídka různých zdrojů dat



Obrázek 4.56: Sisense – nastavení zdroje dat prvního příkladu



Obrázek 4.57: Sisense – editor výrazů

2. Otvírám si editor (viz obr. 4.57), který mi nabídne dostupné fieldy / sloupce a případně i funkce, vystačím si ale s intuitivním použitím aritmetických operátorů. Výsledný výraz je následující: „[one]+[two]+[three]“ (stejnou podobu měl i obdobný výraz u Syncfusion Dashboard).

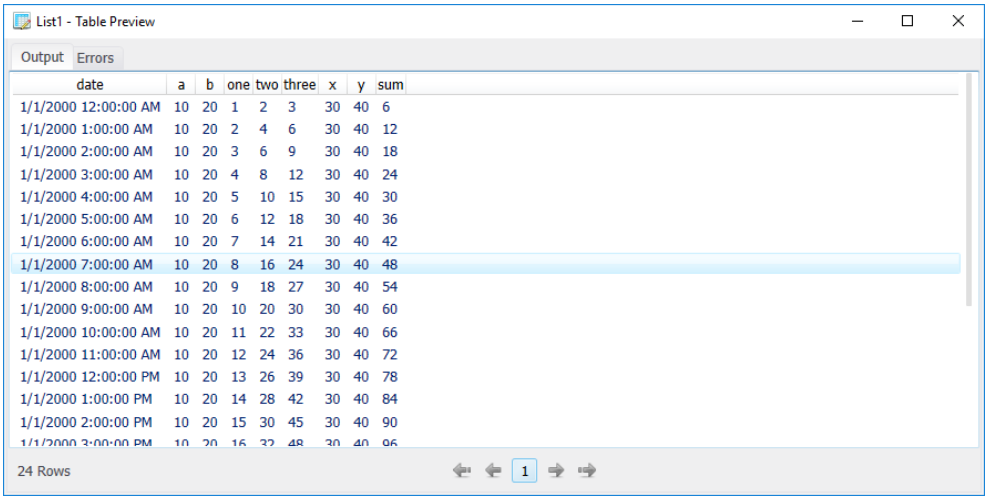
Nyní je potřeba schéma sestavit, aby bylo možné jej použít při tvorbě dashboardu (viz výše). Při sestavení však narážím na komplikace s přihlášením k databázi. Tento problém je znám, nicméně mé pokusy problém řešit podle doporučených postupů selhávají [26]. Připojení k databázi jsem měl pomocí Windows autentizace, namísto té zkouším vytvořit SQL Server autentizaci – tedy pomocí jména a hesla. Při sestavení však nastává stejný problém. Nakonec jsem problém vyřešil tím, že jsem testovací data převedl do Microsoft Office Excel souboru a ten jsem použil jako zdroj dat. Sestavení již proběhlo bez problémů.

Náhled na připravenou tabulku dat, který se po sestavení zpřístupnil, se nachází na obrázku 4.58. V náhledu jsou z neznámého důvodu vidět i sloupce, které byly skryty (viz výše).

Zdroj dat pro dashboard (schéma) je tedy připravený a je uložen jako soubor „.ecube“. Tento soubor je binární povahy, podobně jako soubory „.sydx“ u Syncfusion Dashboard. Nenacházím k němu bližší specifikaci a zřejmě tedy není tedy k dispozici doménově specifický jazyk, kterým by byla definice schématu popsána (jako např. v případě SSRS).

Dalším krokem je tvorba dashboardu, pomocí již zmíněného webového rozhraní Sisense. Při tvorbě dashboardu vybírám připravené schéma, ze kterého bude dashboard čerpat data.

Při přidání widgetu namísto výběru widgetu a následného výběru dat (jako u SSRS a Syncfusion Dashboard), nejprve vybírám data, která chci zobrazit, a až výběrem těchto dat se mi omezuje paleta použitelných widgetů. Data tedy definují možné widgety, které lze pro jejich zobrazení použít. Tento přístup je



date	a	b	one	two	three	x	y	sum
1/1/2000 12:00:00 AM	10	20	1	2	3	30	40	6
1/1/2000 1:00:00 AM	10	20	2	4	6	30	40	12
1/1/2000 2:00:00 AM	10	20	3	6	9	30	40	18
1/1/2000 3:00:00 AM	10	20	4	8	12	30	40	24
1/1/2000 4:00:00 AM	10	20	5	10	15	30	40	30
1/1/2000 5:00:00 AM	10	20	6	12	18	30	40	36
1/1/2000 6:00:00 AM	10	20	7	14	21	30	40	42
1/1/2000 7:00:00 AM	10	20	8	16	24	30	40	48
1/1/2000 8:00:00 AM	10	20	9	18	27	30	40	54
1/1/2000 9:00:00 AM	10	20	10	20	30	30	40	60
1/1/2000 10:00:00 AM	10	20	11	22	33	30	40	66
1/1/2000 11:00:00 AM	10	20	12	24	36	30	40	72
1/1/2000 12:00:00 PM	10	20	13	26	39	30	40	78
1/1/2000 1:00:00 PM	10	20	14	28	42	30	40	84
1/1/2000 2:00:00 PM	10	20	15	30	45	30	40	90

Obrázek 4.58: Sisense – náhled na data ve zdroji dat

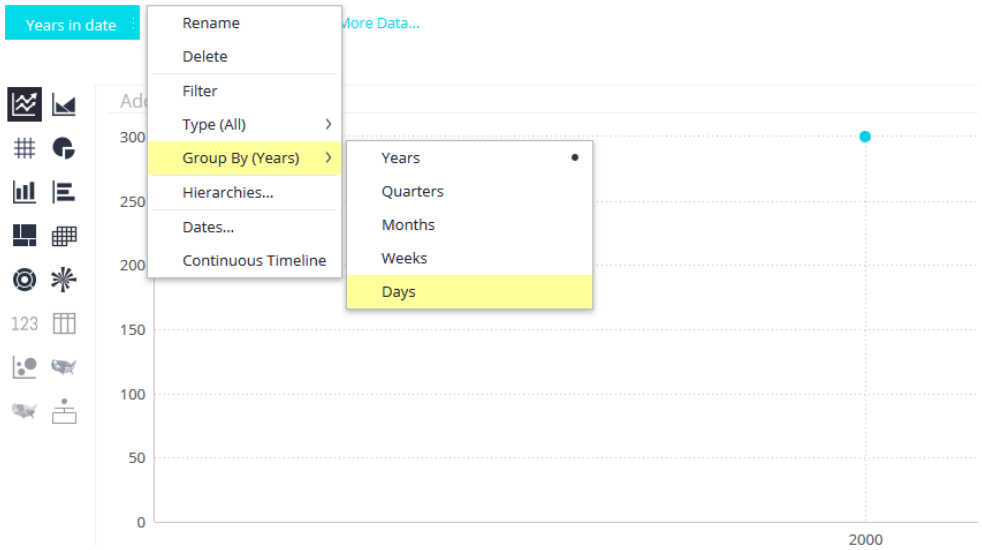
bližší našemu Jazyku, kde také první definuji, co chci zobrazit pomocí datových pohledů a až jako poslední krok volím zobrazovací komponentu, která výsledný pohled zobrazí.

Při přidání sloupce s daty jsou data seskupena do let, stejně jako v Syncfusion Dashboard, ale na rozdíl od tohoto nástroje zde není možnost změnit seskupení na hodiny, či seskupení v jemnější granularitě. Tím pádem nenacházím možnost zabránit promítnutí sloupce hodnot dat do jedné hodnoty pro rok 2000 (viz obr. 4.59). Zdá se, že software nepočítá s prací s daty s jemnější granularitou než jeden den. Dokonce ani v nastavení formátovacího řetězce dat není možné pracovat s jemnější granularitou než dnem (viz obr. 4.60). Dle mého názoru by absence této funkcionality mohla být vysvětlena tím, že widgety typicky zobrazují data delšího období jako např. měsíců souhrnně a v těchto souhrnech ve většině případů zřejmě není potřeba zacházet do příliš-ného detailu, jako do jednotlivých hodin.

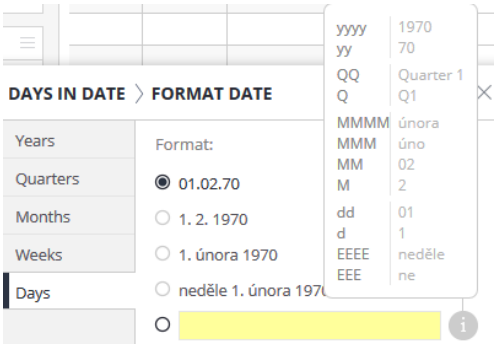
Abych této nežádoucí agregaci dat předešel, provedl jsem úpravu v testovací sadě dat, kde jsem namísto sloupce s daty umístil sloupec s číslem hodiny, datového typu integer.

Teď již úspěšně vybírám potřebná data a widget spojnicového grafu (viz obr. 4.61). Tabulka se tvoří identicky. Dashboard je tak hotový a stejně jako u předešlých nástrojů je možné zobrazit jeho náhled (View Mode, pro návrh dashboardu je zde pak Design Mode). Zobrazení ve View Mode i v Design Mode je téměř shodné (viz obr. 4.62 a 4.63) – i při návrhu jsou tedy již vidět skutečná data, na rozdíl od nástroje SSRS.

Výsledný dashboard je možné podobně jako u předchozích nástrojů exportovat. Exportované dashboardy ve formátu PDF tohoto i následujícího příkladu se nachází na přiloženém DVD.



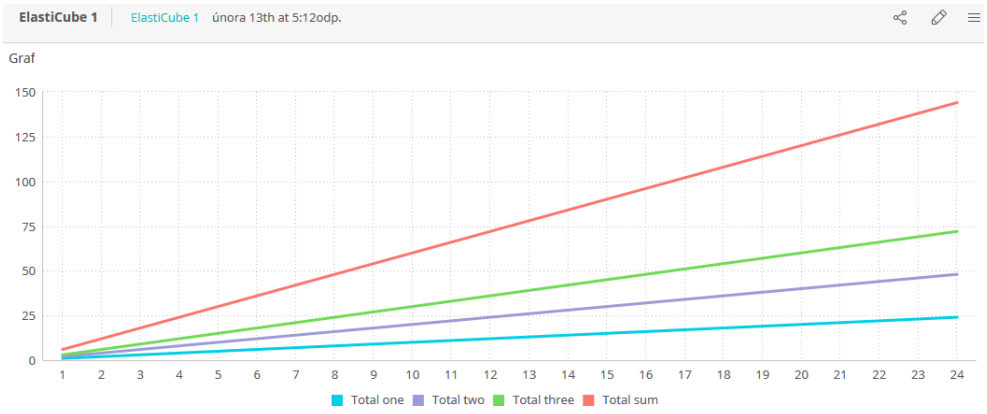
Obrázek 4.59: Sisense – absence podpory hodinové granularity při zobrazení datumu



Obrázek 4.60: Sisense – absence podpory hodinové granularity při formátování datumu

Obrázek 4.61: Sisense – napojení grafu na data prvního příkladu

Obrázek 4.62: Sisense – náhled na graf prvního příkladu v Designer režimu



Tabulka

hour	one	two	three	sum
1	1	2	3	6
2	2	4	6	12
3	3	6	9	18
4	4	8	12	24
5	5	10	15	30

Obrázek 4.63: Sisense – náhled na graf prvního příkladu ve výsledném dashboardu

4.4.2 Příklad na datové operace 2

Druhý příklad podobně jako první testuje funkcionalitu našeho Jazyka a datových pohledů v tomto nástroji. Jedná se o příklad použitý při testování nástroje SSRS a Syncfusion Dashboard a jeho bližší popis je k dispozici v podkapitole 4.1.13 u průzkumu SSRS.

Začnu tvorbou schématu. Kvůli předpokládané komplikaci s nemožností seskupování dat v hodinové granularitě, stejně, jako u příkladu 1, nahrazuji v testovací sadě dat sloupec s daty za sloupec s čísly hodin.

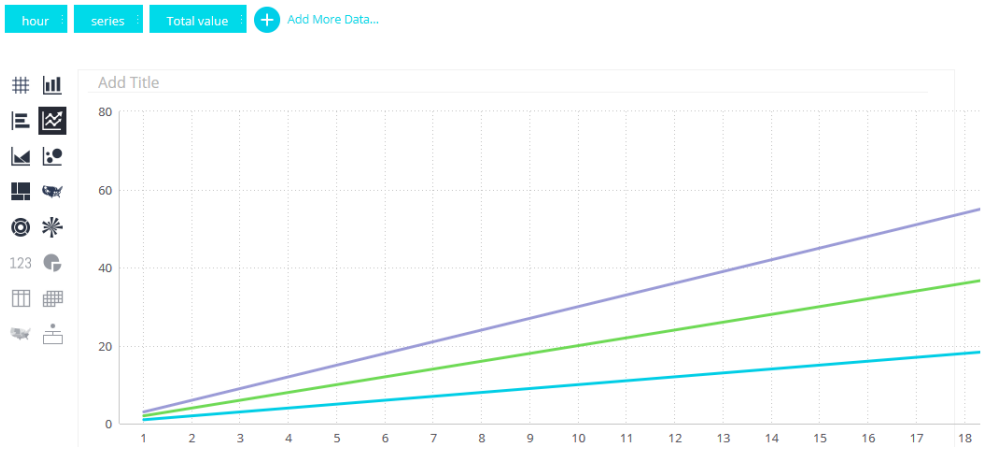
Po napojení na testovací databázi ponechávám z vybrané tabulky všechny sloupce – *series*, *series_category*, *hour* a *value*.

Seskupení dat po časových řadách nebo jejich kategoriích při přípravě schématu nenacházím a tak jej odkládám až do fáze přípravy dashboardu (stejně jako u Syncfusion Dashboard).

Po sestavení schématu tvořím na jeho základě dashboard.

Nejprve přidám do dashboardu graf. Při přidání sloupců *hour* a *series* bez agregace a sloupce *values* s agregací součtu se v grafu zobrazují jednotlivé časové řady. Po sloupcích *hour* a *series* tedy proběhlo seskupení (viz obr. 4.64). Při přidání sloupců *hour* a *series* bez agregace a sloupce *values* s agregací součtu se pak v grafu zobrazují součty časových řad po kategoriích.

Opět se mi však nedaří dostat do grafu jak jednotlivé časové řady, tak jejich součty po kategoriích – podobně jako u SSRS a Syncfusion Dashboard.



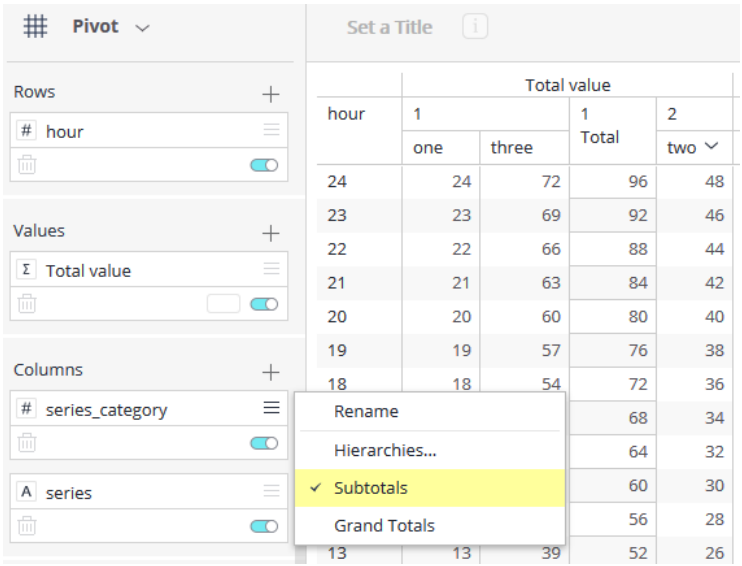
Obrázek 4.64: Sisense – napojení grafu na data v druhém příkladu

Je zde však ještě pokročilé nastavení grafu (záložka Advanced Configuration v lévé dolní části obr. 4.61) – zde je rozdělení zdrojových sloupců do kategorií *X-Axis*, *Values* a *Break by*. V *X-Axis* se nachází sloupec *hour*, ve *Values* se nachází sloupec *values* a v *Break by* sloupec *series* pro první graf nebo *series_category* pro druhý graf. Potřeboval bych přidat do kategorie *Break by* jak sloupec *series*, tak *series_category*, což nelze. Abych v grafech zobrazil požadovaná data, přidávám tedy do dashboardu dva grafy, jeden s jednotlivými řadami a druhý se součty řad po kategoriích (podobně jako u předchozích nástrojů).

Dále chci přidat widget tabulky, pomocí základního výběru dat se mi data vhodně seskupit nedaří a postupuji tedy stejně jako u grafu do pokročilého nastavení (viz obr. 4.65). Zde musím přiřadit zdrojové sloupce do tří kategorií – *Rows*, *Values* a *Columns*. Kategorie se nazývají jinak než u grafu, pro zobrazení dat tohoto příkladu jsou však kategorie identické a to tak, že:

- kategorie X-Axis u grafu odpovídá kategorii Rows u tabulky,
- kategorie Values u grafu odpovídá kategorii Values u tabulky,
- kategorie Break by u grafu odpovídá kategorii Columns u tabulky.

V tabulce lze buď zobrazit jednotlivé časové řady, nebo jejich součty po kategoriích, stejným přiřazením zdrojových sloupců do kategoriích jako v grafu. Když však do kategorie *Rows* přidávám oba sloupce *series* i *series_category* (podobně jako jsem to úspěšně udělal v Syncfusion Dashboard), v popiscích sloupců jsou sice vidět skupiny, určující jaká kategorie obsahuje jaké časové řady, ale není zde sloupec se součtem kategorie, který jsem chtěl získat. V nastavení sloupce *series_category* však nacházím možnost *Subtotals* (viz obr.



Obrázek 4.65: Sisense – pokročilé nastavení tabulky

4.65), která sloupce se součty po kategoriích dodá (u druhé kategorie sloupec není, protože je v ní pouze jeden sloupec).

Výsledný dashboard lze vidět obrázku 4.66.

Dále se zaměřím na pojmenování agregačních sloupců, aby bylo možné použít smysluplnější názvy než jen čísla kategorií časových řad u součtů po kategoriích. V tomto nástroji však nenacházím možnost, jak to udělat ani v grafech, ani v tabulkách. Tato zkoumaná funkcionalita, která je přítomna i u našeho Jazyka, se mi u předchozích nástrojů podařila realizovat.

Funkcionalitu rozbalovacích skupin sloupců v tabulkách, která je v našem Jazyce, jsem v tomto nástroji také nenašel.

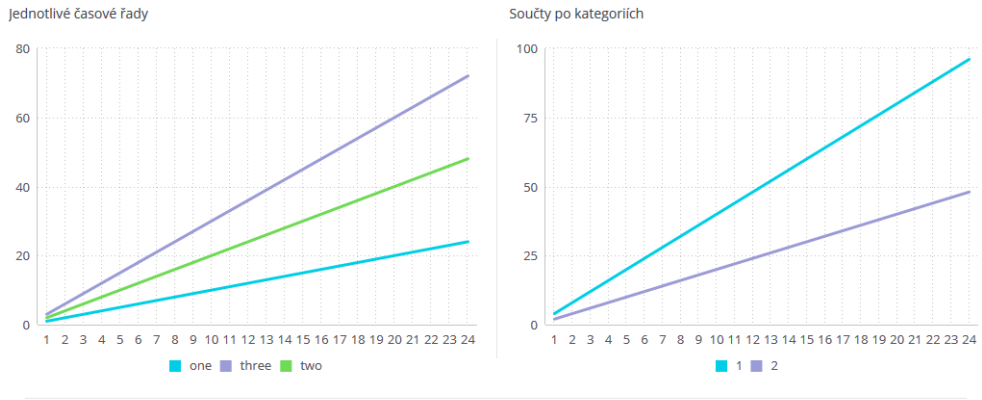
4.5 Souhrn poznatků z průzkumu

Následuje souhrn poznatků z jednotlivých nástrojů a závěrem pak srovnání výhod a nevýhod těchto nástrojů a z nich přebraná inspirace do vlastního nástroje.

4.5.1 SSRS – Report Builder

Nástroj SSRS generuje předpisy v doménově specifickém jazyce RDL. RDL obsahuje řadu podobností s naším Jazykem a stejným způsobem i náš nástroj generuje Předpis v našem Jazyce.

Značná část zaměření nástroje je na vzhled reportu, tedy rozmístění a nastavení zobrazovacích komponent zprávy. To se pak projevuje i ve složitosti



Oboje

hour	Total value 1 one	1 three	1 Total	2 two
24	24	72	96	48
23	23	69	92	46
22	22	66	88	44
21	21	63	84	42
20	20	60	80	40
19	19	57	76	38
18	18	54	72	36

Obrázek 4.66: Sisense – výsledný dashboard druhého příkladu

RDL předpisu, kde se většina jeho obsahu zabývá právě tímto. Jak již bylo řečeno, náš Jazyk se tímto zabývá jen okrajově – rozmístění zobrazovacích komponent je pevně dáno šablonou aplikace. V Jazyce se pak pouze řeší kolik grafů a tabulek v šabloně bude a dále jim lze parametrizovat vzhled, ale pouze do omezené míry. Primárně se pak Jazyk věnuje zpracování dat, které se mají v datagridech a grafech zobrazit.

Vygenerované RDL předpisy jsou velmi složité i pro jednoduché reporty. Jejich ruční tvorba by zřejmě byla ještě méně udržitelná, než jak je to u našeho Předpisu. Je to zřejmě dáno tím, že předpis je vygenerován velmi obecně i v případě, že report této obecnosti nevyužívá a dále zřejmě také neprobíhá žádné zjednodušení, jako odstranění nevyužitých elementů apod. Pokud je předpis určen především pro vizuální editaci v GUI, pak tento fakt asi není negativum. V našem nástroji vygenerované Předpisy obsahují také některou obecnost navíc, která by mohla být odstraněna v případě jednoduchých reportů. Celkově však jsou dle mého názoru naše vygenerované Předpisy jednodušší a přehlednější (viz podkapitoly 3.2 pro náš předpis a 4.1.11 pro RDL předpis). Je to samozřejmě z velké části dáno tím, že náš nástroj se zabývá užší problematikou než SSRS. Důsledkem nadměrné obecnosti RDL je pak to, že je někdy velmi obtížné se vyznat se ve velkém množství elementů a najít co hledám. Jak již bylo diskutováno, je zde totiž mnoho prázdných elementů a dále je většina elementů hluboce zanořená (hloubka je až v desítkách zano-

ření), přitom po několika zanořeních je zde často jen jeden nebo několik málo elementů. Definice je tak zbytečně složitá, i když popsaný report složitý není a člověk se musí naučit ignorovat spoustu „šumu“ v definici, aby našel to, co je podstatné.

Předpis v RDL může být rozdělen na části (Report parts), které lze znovupoužít v dalších reportech. Tento přístup umožňuje omezit duplikaci kódu. V našem jazyce lze podobným způsobem znovupoužít, resp. vkládat stromy datových pohledů do jiných stromů a tímto způsobem lze omezit duplikaci při opakovaném způsobu zpracování dat a vyčlenit jej na jedno místo.

Při editaci reportu jsou data v nekonkrétní podobě – je zde vidět pouze jejich struktura. Při spuštění reportu potom jsou vidět skutečná data. Je zde příkladový počet a hodnoty dat – např. Full Name 1 namísto John Smith apod. V našem nástroji se při definici Předpisu používají také pouze informace o struktuře dat, nezávisle na jejich konkrétních hodnotách. Předpisy jsou totiž tvořeny pro datové třídy a v této fázi není známo, jaké hodnoty budou v jednotlivých instancích. Je zde však také možnost zobrazit v dané části zpracování, resp. pro každý vytvořený datový pohled, náhled na data, kde uživatel přímo v nástroji vidí, co předpis dělá s konkrétními daty (viz podkapitola 5.2). V tomto náhledu se zobrazí konkrétní data k vybranému datovému objektu z této třídy.

V SSRS probíhá značná část přípravy dat až v nastavení dat dané zobrazovací komponenty. Kvůli tomu se duplikuje část zpracování dat při zobrazení ve více komponentách. Datové pohledy tento problém eliminují, neboť popisují zpracování dat nezávisle na zobrazovací komponentě – lze je pak použít jak v grafech, tak v datagridech.

Query Designer je obsažený nástroj v SSRS, který se zabývá přípravou dat, resp. dotazem, který data zpracovává. Tento nástroj nabízí i definici složitějších operací nad daty, jako jejich agregace a seskupení. Stromy datových pohledů však dle mého názoru nabízí větší variabilitu při přípravě dat. Nesetkáváme se zde např. s problémy, že by v jednom datovém pohledy nemohla být ta samá data vícekrát, pouze v jiné formě, resp. jednou agregované a podruhé nikoli.

Pojmenování agregovaných sloupců lze podobně jako v Jazyce realizovat pomocí předpisu názvu sloupců, nicméně pro dosažení požadované podoby pojmenování byl nutný až programátorský zásah (předpis s využitím znalosti Visual Basic ve formě „="Součet kategorie " + Fields!series_category.Value-.ToString()“). V našem nástroji lze do řetězců s názvem vkládat hodnotu parametru dané skupiny intuitivnějším způsobem (viz dále v podkapitole 4.5.4). Předpis názvu v SSRS je však obecnější, nicméně tato míra obecnosti není v našem nástroji v současné době potřeba.

V nástroji jsem se setkal s jeho neodladěností v určitých situacích. Poprvé to byla neošetřená/uživatelsky nesrozumitelná chyba související s oddělovačem desetinných hodnot v souvislosti se zvolenou českou lokalizací ve Windows. Později, při navázání dat a popisků buněk v tabulce, nástroj také vyhazoval

neošetřené chyby.

Realizace prvního srovnávacího příkladu se zdařila. To, co u mě realizoval datový pohled pro výběr, se v nástroji provedlo zaškrtáním sloupců v Query Designeru, který vytvořil příslušný SQL dotaz. Datový pohled pro agregaci – pro součet sloupců, pak byl vytvořen až v grafu / v tabulce pomocí aritmetické operace ve výrazu. Funkce datového pohledu pro sloučení dat se přirovnává hůře, neboť výsledné sloupce bylo potřeba vybrat až v zobrazovací komponentě.

Realizace druhého srovnávacího příkladu se také, až na pár detailů, zdařila. Druhý příklad byl komplikován složitějším mapování časových řad na relační databázi. Je otázkou, jestli by nebyl vhodnější jiný typ databáze, který by umožňoval přímočařejší práci s daty povahy časových řad. Po odsunutí operací s daty z SQL dotazu, kde to nebylo příliš použitelné (dataset obsahoval pouze jeden sloupec a zobrazovací komponenta tím pádem také), do nastavení zobrazovacích komponent, bylo nakonec přijatelně obtížné pokročilé operace s daty provést. Nepodařilo se mi sice vyladit drobnosti jako sloučení obou grafů (s jednotlivými a s agregovanými průběhy) do jednoho a umístění popisků tabulky na stejný řádek, ale kromě toho, se zdařilo nasimulovat tyto složitější operace prováděné datovými pohledy pomocí testovaného nástroje SSRS.

4.5.2 Syncfusion Dashboard

Nástroj je moderního vzhledu a obsahuje zabudované nápovědy a tipy na intuitivních místech v aplikaci (což u nástroje SSRS nebylo).

Syncfusion nabízí intuitivní a přehlednou přípravu zdroje dat, možnost definice součtových sloupců popsaných aritmetickým výrazem. Nástroj dále obsahuje výrazy (expressions) podobně jako v SSRS a pomocí nich je možné definovat sloupce vzniklé provedením operací nad zdrojovými sloupci (podobně jako v našem Jazyce datový pohled pro agregace).

Podobně jako u SSRS je značná část přípravy dat až v nastavení zobrazovací komponenty. Toto nastavení a přiřazení dat je však dle mého názoru řešeno přehlednější formou než u SSRS.

Data jsou již při přípravě widgetu vidět v konkrétní podobě, na rozdíl od SSRS. Podle mého názoru by z principu i v SSRS mohla být v designeru vidět skutečná data pro lepší představu.

Nástroj se podobně jako SSRS zabývá více než náš nástroj rozložením a vzhledem celého dashboardu.

Předpis pro dashboard je pak uložený v souboru, tento soubor však je binární povahy a nelze tak nahlédnout do doménově specifického jazyka, jako tomu bylo u SSRS a u našeho nástroje.

Zdařilá byla tvorba skupin sloupců v tabulce při seskupení hodnot, podobně jako to lze v našem nástroji. Tímto způsobem se mi to však u SSRS nepodařilo. Tato tvorba skupin však probíhá automaticky při seskupení dat

v tabulce a nenašel jsem způsob, jak ji v nástroji editovat. Snažil jsem se toto nastavení najít a upravit ho nebo ho jen vypnout, v nastavení tabulky jsem však žádný parametr, který by seskupení ovlivňoval, nenašel. Dále jsem zkoušel ručně nastavit skupiny sloupců i pro první příklad, kde seskupení není (tak jak to libovolně lze v našem Jazyce v datagridu), ale ani zde jsem tuto možnost nenašel. Dle mého názoru je poněkud zvláštní, pokud tabulka nenabízí v tomto nástroji parametrizaci těchto relativně základních vlastností.

Předpis pro pojmenování agregačních sloupců je řešen výrazem, podobně jako u SSRS. Jeho editace je však jednodušší a náš nástroj má tuto funkcionalitu udělánu velmi podobně. Tento předpis se zdařil nadefinovat u grafu, nicméně u tabulky jsem tuto funkcionalitu nenašel.

Příprava obou srovnávacích příkladů se zdařila a probíhala podobným způsobem jako u SSRS. Drobné problémy jsou opět se zobrazením stejných dat dvakrát ve stejném widgetu, jednou v agregované a podruhé v původní podobě. Dále jsou zde drobné nedostatky v absenci nastavení skupin sloupců v tabulce.

4.5.3 Sisense

V tomto nástroji je oddělená příprava zdroje dat dashboardu od tvorby dashboardu – tedy určení rozložení widgetů a jejich napojení na data. To je podobný princip jako v našem nástroji, kde je nejprve potřeba definovat datové pohledy a až potom se určí, kde budou zobrazeny.

V přípravě zdroje dat lze podobně jako v Syncfusion vytvářet nové sloupce předepsané aritmetickým výrazem ze zdrojových sloupců, podobně jako agregační datové pohledy v našem nástroji.

Soubory, ve kterých je předpis zpracování dat, jsou binární povahy, podobně jako v Syncfusion, takže není možné je prohlížet v doménově specifickém jazyce, jako tomu bylo u SSRS.

Při přidání widgetu namísto výběru widgetu a následného výběru dat (jako u SSRS a Syncfusion Dashboard), nejprve vybírám data, která chci zobrazit, a až výběrem těchto dat se mi omezuje paleta použitelných widgetů. Data tedy definují možné widgety, které lze pro jejich zobrazení použít. Tento přístup je bližší našemu Jazyku, kde také první definuji, co chci zobrazit pomocí datových pohledů a až jako poslední krok volím zobrazovací komponentu, která výsledný pohled zobrazí.

V nástroji narážím na problém zobrazit data po jednotlivých hodinách, neboť nejjemnější dostupná časová granularita je jeden den. Dokonce ani v nastavení formátovacího řetězce dat jsem nenalezl způsob, jak pracovat s jemnější granularitou než dnem. Dle mého názoru by absence této funkcionality mohla být vysvětlena tím, že widgety typicky zobrazují data delšího období jako např. měsíců souhrnně a v těchto souhrnech ve většině případů zřejmě není potřeba zacházet do přílišného detailu, jako do jednotlivých hodin. Nakonec jsem se tomuto problému vyhnul tím, že jsem upravil testovací data, tak aby

Tabulka 4.4: Výhody prozkoumaných nástrojů

Nástroj	Výhody
SSRS – Report Builder	1. Dostupné DSL s předpisem pro zpracování dat. 2. Obecnost výrazů – lze použít Visual Basic a různé knihovny. 3. Větší obecnost nástroje.
Syncfusion Dashboard	1. Zabudovaná nápověda na vhodných místech. 2. Moderní a intuitivní GUI. 3. Tvorba nových sloupců na základě aritmetického výrazu již při tvorbě zdroje dat.
Sisense	1. Výběr zobrazených dat před výběrem zobrazovací komponenty. 2. Moderní a intuitivní GUI. 3. Tvorba nových sloupců na základě aritmetického výrazu již při tvorbě zdroje dat.

nebyla hodinová a realizaci srovnávacích příkladů v tomto nástroji to tedy dále nebránilo.

Stejně jako u Syncfusion již při návrhu dashboardu jsou vidět data ve skutečné podobě. Tento náhled na skutečná data je důležitý pro lepší představu o dashboardu při jeho tvorbě a je hlavní inspirací z těchto nástrojů. Na základě tohoto poznatku vznikl v našem nástroji náhled na data jednotlivých datových pohledů při jejich tvorbě. Bez tohoto prvku by byla tvorba uživatelských schémat v našem nástroji velmi abstraktní.

Realizace srovnávacích příkladů se až na několik drobností zdařila. Mezi tyto drobnosti patří znovu problém zobrazit v jednom grafu zároveň stejná data v podobě agregované a v podobě bez agregace. Na tento problém jsem narazil ve všech testovaných nástrojích. Co se týče seskupování sloupců v tabulce, tak jsou zde sloupce automaticky seskupeny, pokud jsou zobrazeny v tabulce původní a agregovaná data. Tyto skupiny však nelze rozbalit a sbalit jako v ostatních a v našem nástroji, ale skupiny jsou dány pouze vzhledem hlavičky tabulky.

Celkově na mě nástroj působil moderním a uživatelsky přívětivým dojmem, podobně jako Syncfusion Dashboard.

4.5.4 Výhody a nevýhody

Závěrečná podkapitola průzkumu shrnuje výhody, nevýhody a další rysy zkoumaných nástrojů a dále popisuje dopad na návrh a implementaci vlastního nástroje.

V tabulkách jsou vypsány výhody 4.4 a nevýhody 4.5 jednotlivých nástrojů, v souvislosti s touto prací.

Dalším pozitivem (mimo výhody v tabulce), které mají všechny nástroje

Tabulka 4.5: Nevýhody prozkoumaných nástrojů

Nástroj	Nevýhody
SSRS – Report Builder	1. Data při návrhu nejsou v konkrétní podobě. 2. Neintuitivní tvorba výrazu pro pojmenování agregačních sloupců. 3. Neošetřené chyby v aplikaci.
Syncfusion Dashboard	1. Nedostupné DSL s předpisem pro zpracování dat.
Sisense	1. Absence rozbalovacích skupin v tabulkách 2. Nemožnost zobrazit časovou osu jemněji než po dnech. 3. Nedostupné DSL s předpisem pro zpracování dat.

je možnost zobrazit náhled na výslednou podobu zprávy s konkrétní podobou dat. Naopak negativem všech nástrojů bylo, že část zpracování dat probíhá až v zobrazovacích komponentách, což při zobrazení stejných dat v různých komponentách vede na duplikaci.

Inspiraci z průzkumu si náš nástroj bere v několika ohledech:

- Při tvorbě datových pohledů je zde přidán náhled na konkrétní data datových pohledů. Pro libovolný vybraný datový pohled lze zobrazit data k právě vybrané instanci datové třídy v aplikaci.
- Uživatelské rozhraní, pro tvorbu předpisu názvu agregačních sloupců a vkládání zástupných symbolů do něj, je inspirováno nástroji Syncfusion Dashboard a Sisense, kde je pod předpisem možnost vložit relevantní zástupný symbol do tohoto výrazu a jednoduše jej tak spojit se zbytkem řetězce.
- Dále je zde inspirace z nástroje Sisense v myšlence, že příprava a výběr dat předchází výběru zobrazovací komponenty. Datové pohledy popisující zpracování dat jsou vytvářeny jako první a až jako poslední krok je přidání zobrazovacích komponent do schématu.

V souvislosti se zmíněnými rysy prozkoumaných nástrojů náš nástroj generuje interně vyvinuté DSL s předpisem pro zpracování dat. Dále nástroj také obsahuje editor pro tvorbu rozbalovacích skupin v datagridu. A co se týče vlastních sloupců popsaných aritmetickým výrazem, tak v našem nástroji lze, pomocí datových pohledů pro agregaci (které jednotlivě reprezentují jednu aritmetickou operaci) a jejich skládání, vytvořit sloupce odpovídající aritmetickým výrazům, jako tomu bylo v ostatních nástrojích.

Kapitola 5

Návrh a implementace vlastního nástroje

Následující kapitola se bude zabývat implementací vlastního nástroje, který je hlavním výstupem této diplomové práce. Pro první seznámení bude nejprve nástroj demonstrován na příkladech, a až poté budou funkcionalita nástroje a jeho implementace podrobněji rozebrány.

5.1 Ukázka nástroje na příkladech

Tato podkapitola čtenáře provede realizací srovnávacích příkladů v implementovaném nástroji. Jedná se o příklady, které byly popisovány v rešerši existujících nástrojů. Cílem této podkapitoly je, aby čtenář získal lepší představu o nástroji, před jeho podrobným popisem, který je tématem příští podkapitoly.

5.1.1 Příklad na datové operace 1

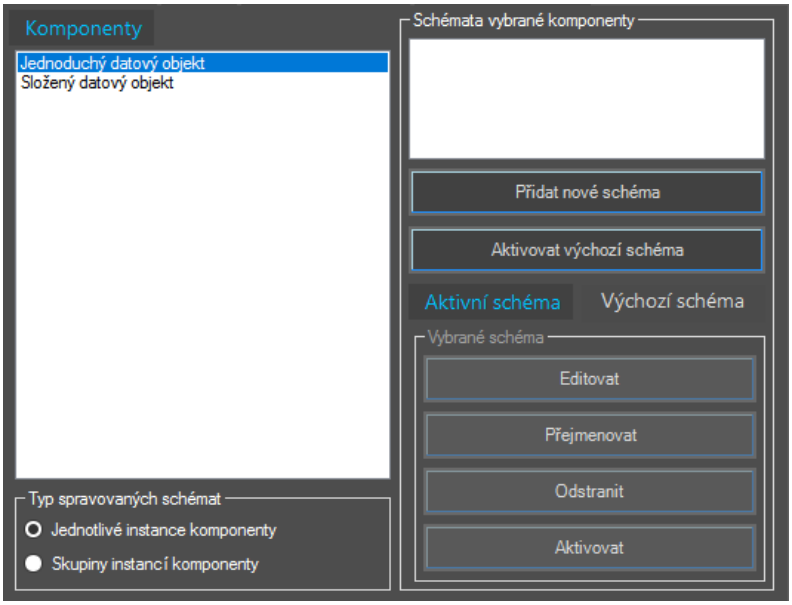
Následující příklad se zabývá vyzkoušením základní funkcionality datových pohledů, jako výběr sloupců, jejich součet a slučování tabulek. Jedná se o stejný příklad jako u všech zkoumaných nástrojů – viz podkapitola 4.1.9 u nástroje SSRS. Ve zmíněné podkapitole se nachází popis a vysvětlení příkladu a před čtením dále je vhodné si jej přečíst.

Ve zkratce, cílem příkladu je, ze zdrojové tabulky, která obsahuje sloupce A, B, 1, 2, 3, X a Y, vybrat pouze sloupce 1, 2, 3 a vytvořit součtový sloupec 1+2+3 a zobrazit ho spolu s nimi.

Pro příklad byla vytvořena datová třída „Jednoduchý datový objekt“, která obsahuje všechny časové řady zdrojové tabulky (A, B, 1, 2, 3, X a Y) jako jednotlivé properties.

Zatím jsem pro datovou třídu nedefinoval zobrazovací schéma, a když se tedy pokusím pro její instanci zobrazit data, použije se výchozí zobrazovací

Obrázek 5.1: Zobrazení výchozího schématu k prvnímu příkladu

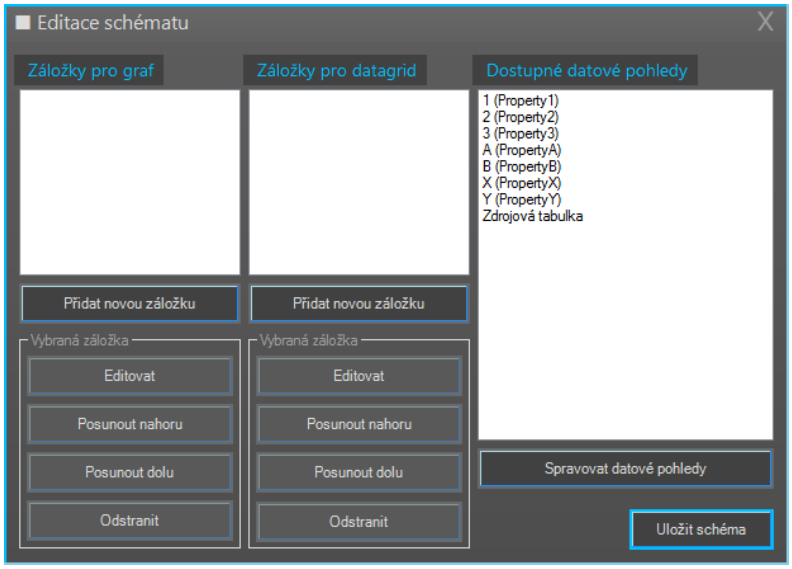


Obrázek 5.2: Správa schémat k prvnímu příkladu

schéma, které pouze zobrazí všechny její časové řady do jedné záložky datagridu (viz obr. 5.1).

Pomocí mého nástroje tedy vytvořím požadované zobrazovací schéma. Na úvodní obrazovce nástroje (viz obr. 5.2) zvolím ze seznamu dostupných datových tříd (neboli komponent) datovou třídu „Jednoduchý datový objekt“ a můžu vidět, že skutečně třída žádná schémata zatím nemá (prázdný seznam vpravo nahoře) a že se používá výchozí schéma (kolonka „Aktivní schéma“). Pomocí tlačítka „Přidat nové schéma“ přidám tedy schéma, příslušně si jej pojmenuji a tlačítkem „Editovat“ přecházím na jeho editaci.

Editace probíhá v novém okně (viz obr. 5.3). Nové schéma nemá žádné záložky v datagridu ani v grafu. Nejprve si však musím připravit data, která



Obrázek 5.3: Editace schématu k prvnímu příkladu

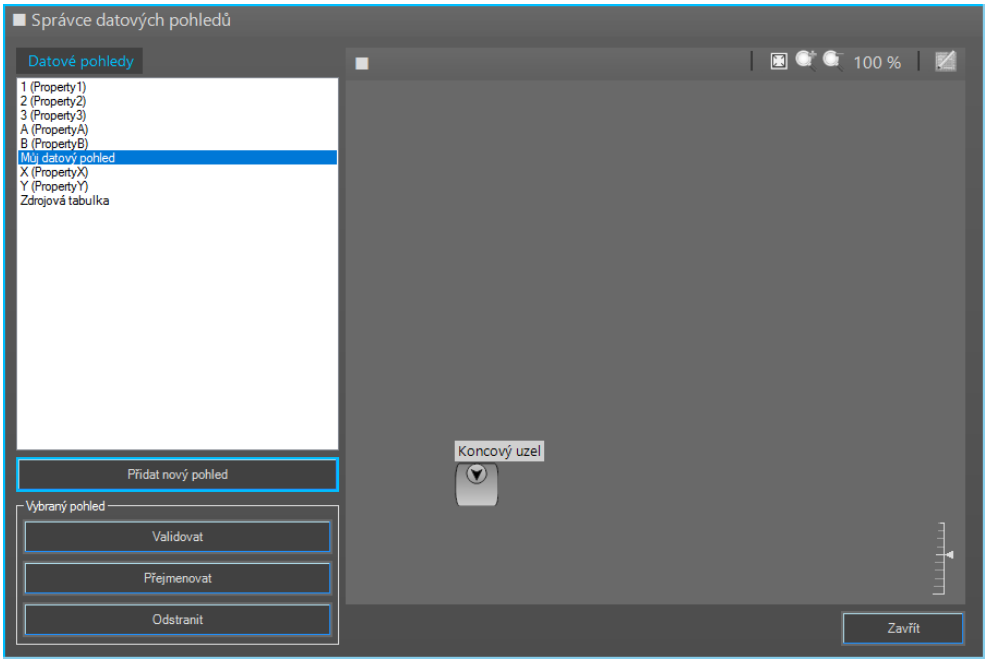
v nich mohu zobrazit, tedy datové pohledy. Jak vidím, tak už některé datové pohledy dostupné mám. Konkrétně zdrojovou tabulku obsahující všechny časové řady a pak pohledy pro každou jednotlivou časovou řadu, resp. její property v datové třídě. Já však chci vytvořit datový pohled, který bude obsahovat sloupce 1, 2, 3 a jejich součet. Proto přecházím tlačítkem „Spravovat datové pohledy“ do správy datový pohledů.

V okně pro správu datových pohledů (obr. 5.4) vidím seznam existujících pohledů a pomocí tlačítka „Přidat nový pohled“, přidávám nový pohled (strom), který provede nad daty požadované operace. V pravé části okna vidím náhled na strom datových pohledů, který je za tímto mým vybraným pohledem skryt. Ve stromu je zatím pouze koncový uzel, ale zatím zde nejsou žádné datové pohledy, ani jejich propojení. Kliknutím na ikonku v pravém horním rohu se tedy přesouvám do editace tohoto stromu pohledů.

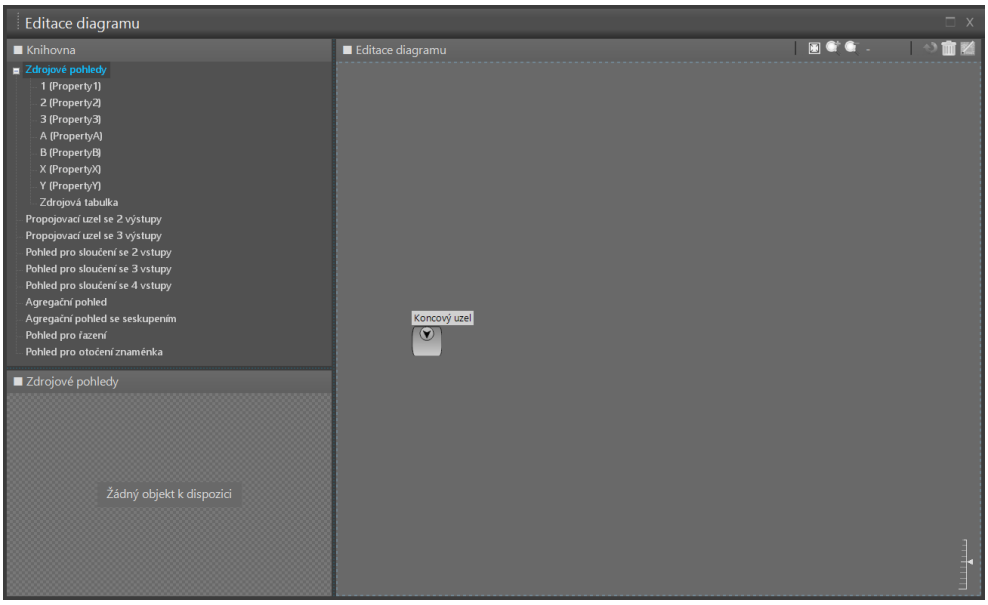
V tomto okně (viz obr. 5.5) budu tvořit strom datových pohledů. V levé horní části vidím dostupné datové pohledy, které mohu použít. V levé dolní části mám potom vybraný pohled, který mohu do svého stromu myší přetáhnout. Tímto způsobem tedy přetáhnu šablony zdrojových pohledů, které budu pro svůj strom potřebovat, tedy pohledy 1, 2 a 3 (a koncový uzel posunu níže, abych si udělal na strom více místa) (viz obr. 5.6).

Pokud bych chtěl zobrazit pouze tyto tři sloupce, potřebuji ještě sloučit tyto pohledy do jednoho, proto přidám pohled pro sloučení se třemi vstupy (viz obr. 5.7).

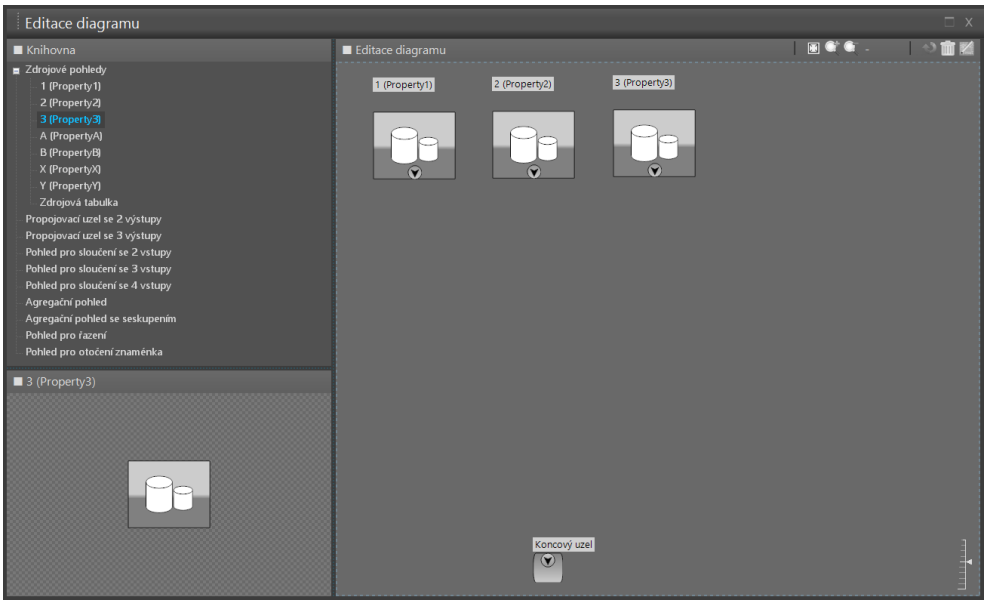
Každý uzel v grafu má vstupní nebo výstupní konektory. Konektory do sebe lze zapojit kliknutím na jeden z nich a přetažením myši na druhý – tím



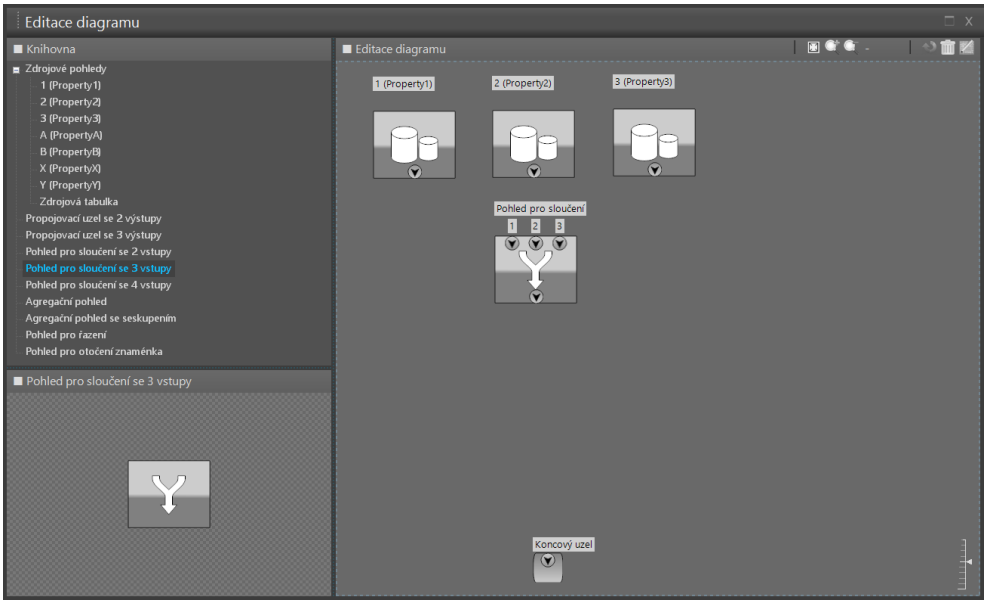
Obrázek 5.4: Správa datových pohledů k prvnímu příkladu



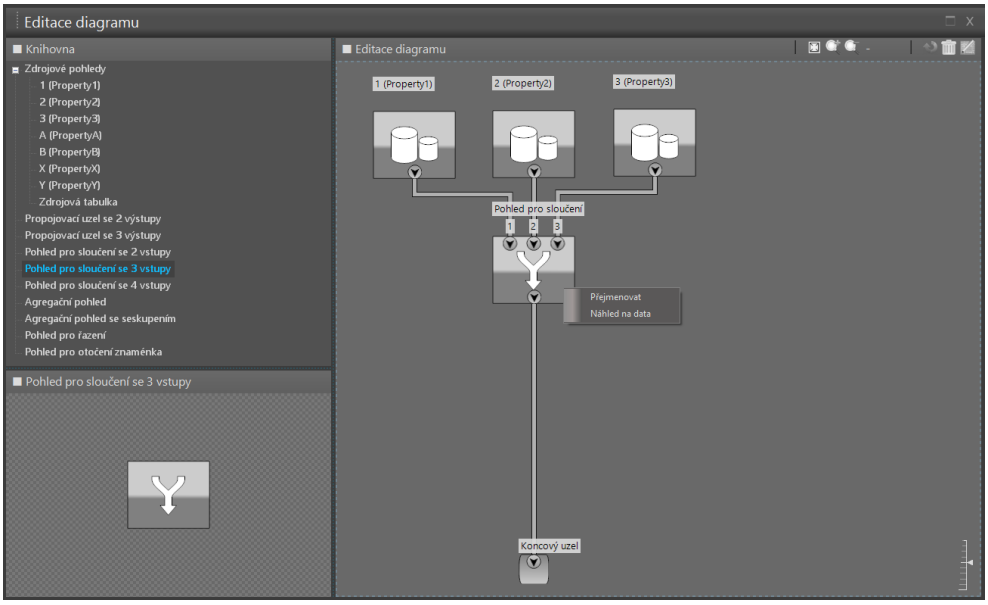
Obrázek 5.5: Editace datových pohledů k prvnímu příkladu – prázdný strom



Obrázek 5.6: Editace datových pohledů k prvnímu příkladu – zdrojové pohledy



Obrázek 5.7: Editace datových pohledů k prvnímu příkladu – sloučení pohledů



Obrázek 5.8: Editace datových pohledů k prvnímu příkladu – propojení základního stromu

se mezi nimi vytvoření propojení. Tímto způsobem tento strom propojím (viz obr. 5.8).

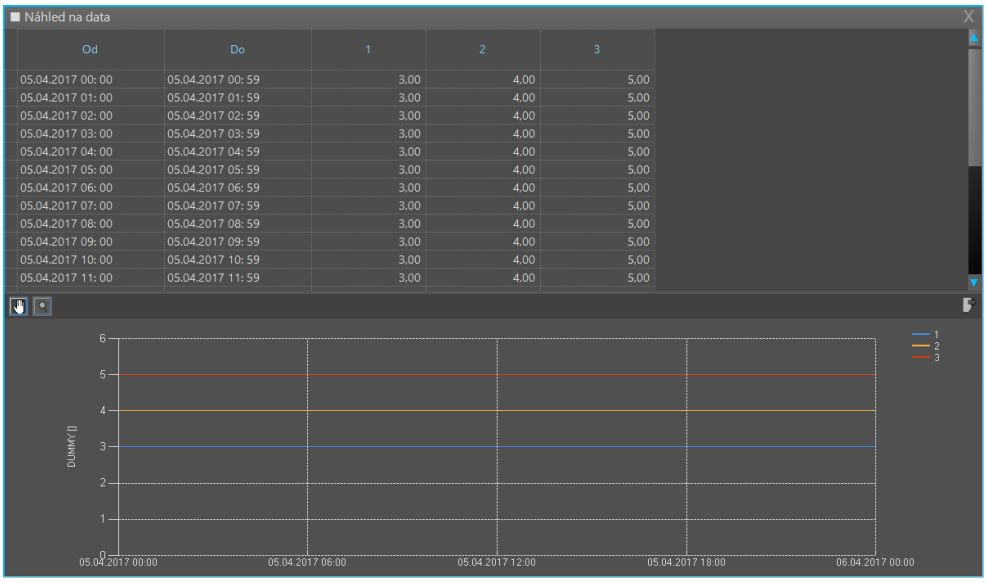
Nyní si mohu ověřit, že mám ve výsledném pohledu, tedy v tom propojeném s koncovým uzlem, požadované sloupce. Přes kontextové menu (viz obr. 5.8) tohoto pohledu zobrazím náhled na data a vidím (viz obr. 5.9), že zatím postupuji správně. Zavřu tedy náhled a pokračuji v tvorbě stromu pohledů.

Abych získal součtový sloupec, přidám agregační pohled (viz obr. 5.10) a přes kontextové menu nastavím jeho parametry ve formuláři (vpravo dole) – nastavím typ agregace na sčítání a pojmenuji výsledný sloupec.

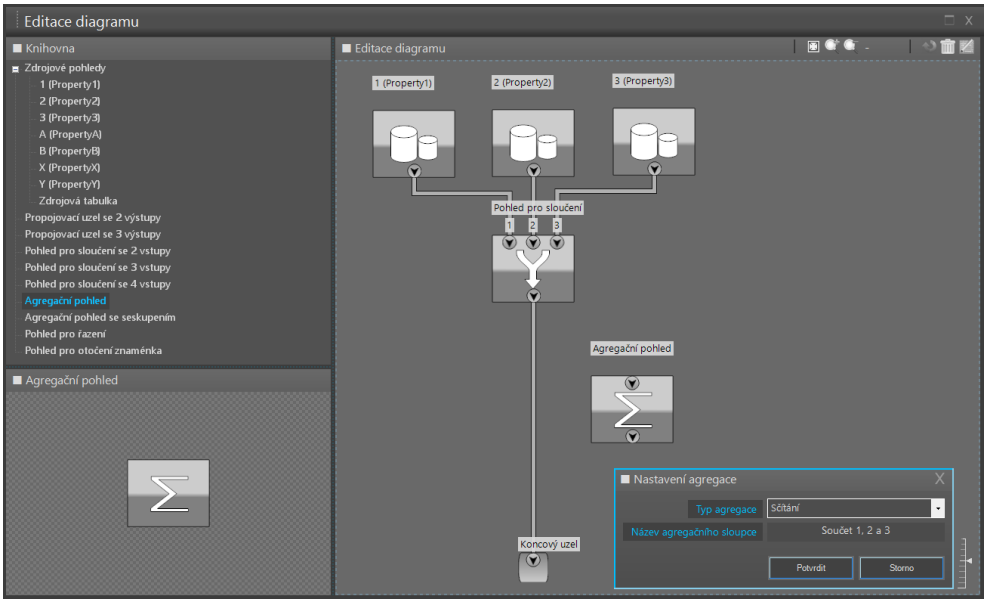
Tento agregační pohled chci začlenit do stromu. Konkrétně chci, aby výsledných pohledem byly jak pohled pro sloučení, tak jeho výstup, na který bude aplikován přidaný agregační pohled. Odstraním nejprve propojení mezi pohledem pro sloučení a koncovým uzlem (označením myší a stisknutím klávesy Delete) a dále přidám propojovací uzel, který mi umožní výstup z pohledu pro sloučení rozdvojit, abych mohl jeden z nich zachovat s původními daty a na druhý z nich aplikovat agregační pohled (viz obr. 5.11).

Na závěr už jen sloučím nezměněný výstup z propojovacího uzlu a výstup z agregačního pohledu stejným způsobem, jako jsem sloučil sloupce 1, 2, 3 (viz obr. 5.12).

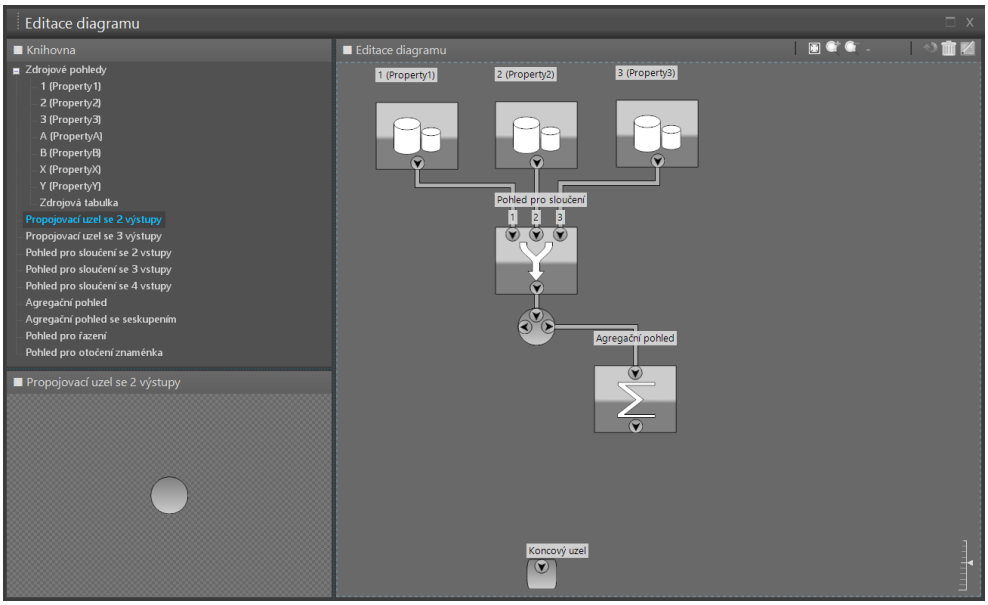
Ukončím tedy editaci stromu pohledů a vracím se zpět na editaci schématu, kde vidím (viz obr. 5.13), že tento můj nový strom je mezi dostupnými pohledy. Dále jej chci zobrazit v záložkách grafu a datagridu – začnu grafem. Přidám novou záložku grafu pomocí tlačítka „Přidat novou záložku“, pojmenuji ji a



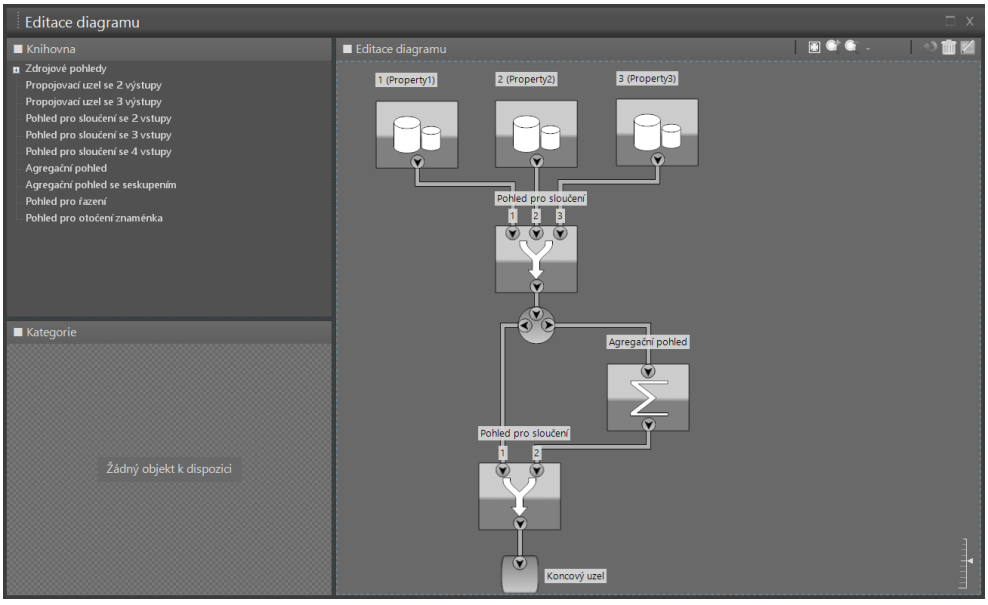
Obrázek 5.9: Editace datových pohledů k prvnímu příkladu – náhled na data



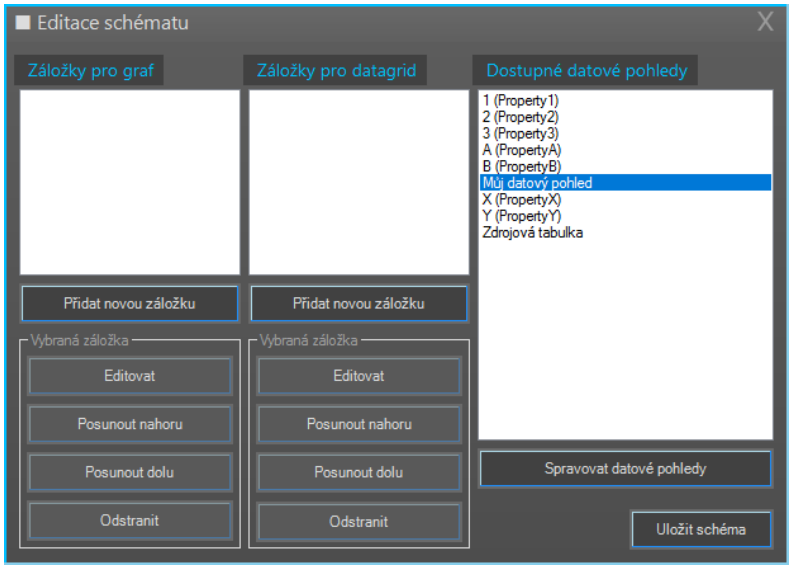
Obrázek 5.10: Editace datových pohledů k prvnímu příkladu – agregační pohled



Obrázek 5.11: Editace datových pohledů k prvnímu příkladu – propojovací uzel



Obrázek 5.12: Editace datových pohledů k prvnímu příkladu – kompletní strom



Obrázek 5.13: Nový strom datových pohledů dostupný k prvnímu příkladu – kompletní strom

pomocí tlačítka „Editovat“ se přesunu na její nastavení, kde nastavím její parametry (viz obr. 5.14). Pojmenuji si záložku, vyberu, aby se v ní zobrazil můj vytvořený datový pohled, a pojmenuji popisek osy Y grafu.

Stejným způsobem přidám záložku datagridu a přesunu se do její editace.

V záložce datagridu (viz obr. 5.15) může být zobrazeno více datových pohledů naráz a také zde lze časové řady datových pohledů dávat do skupin. To však v tuto chvíli nebudu potřebovat, jediné co zatím chci, je zobrazit v datagridu můj vytvořený pohled. Tlačítkem „Vybrat datové pohledy a skupiny“ se přesunu do příslušného nastavení (viz obr. 5.16), kde vyberu datový pohled, potvrdím a vracím se zpět do editace schématu.

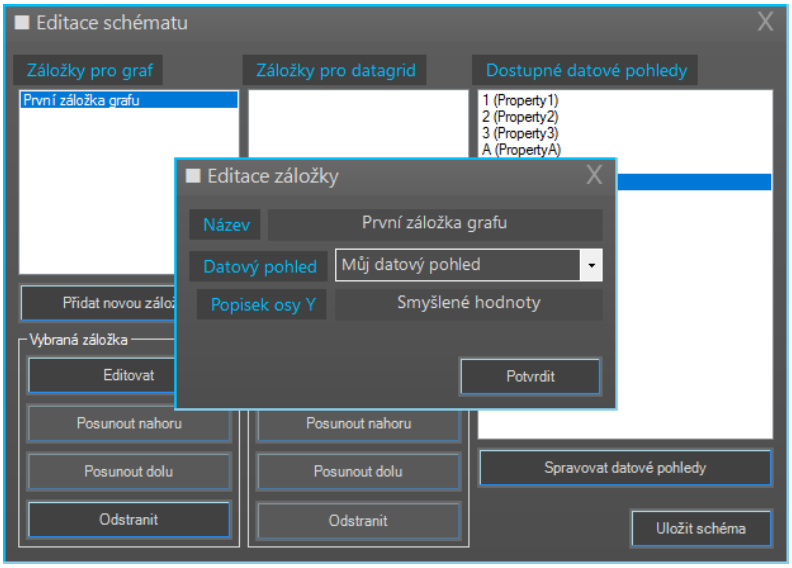
V tuto chvíli mám vše potřebné (viz obr. 5.17), takže mohu uložit schéma (tlačítkem vpravo dole). Před uložením proběhne validace případných chyb v nastavení, ale postupoval jsem správně a tak se uložení zdařilo bez chyb (viz obr. 5.18).

Vyskočím z editace schématu zpět do správy schémat (viz obr. 5.19), kde už jen zbývá nastavit nově vytvořené schéma jako aktivní pomocí tlačítka „Aktivovat“.

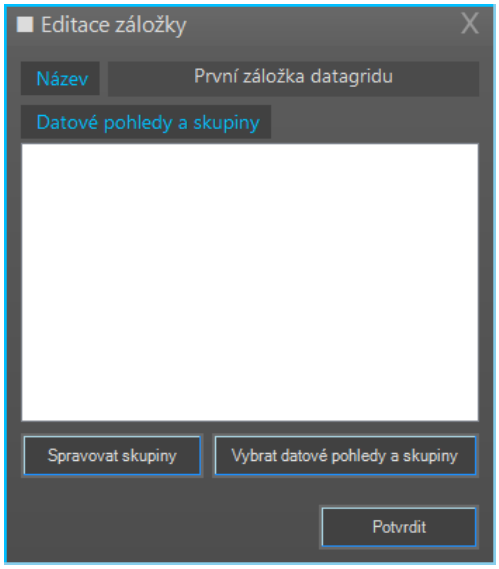
Nyní v aplikaci znovu zobrazím data pro objekt této datové třídy a vidím (viz obr. 5.20), že se data zobrazují tak, jak bylo požadováno.

5.1.2 Příklad na datové operace 2

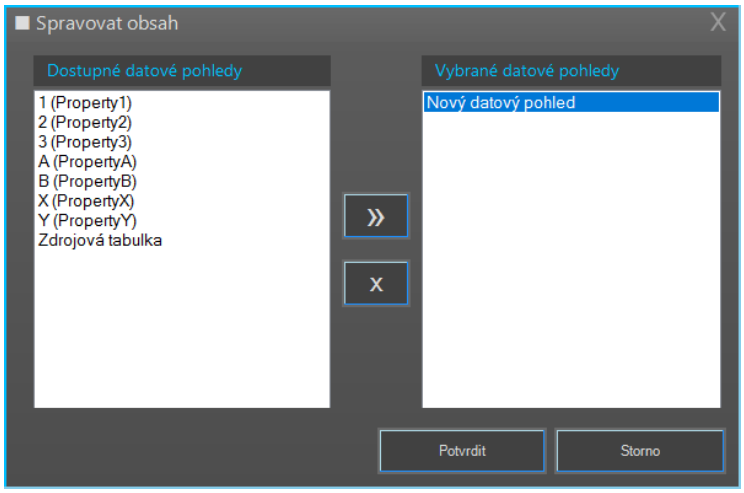
Následující příklad se zabývá vyzkoušením pokročilejší funkcionality datových pohledů, jako agregace se seskupením, řazení a seskupování sloupců v da-



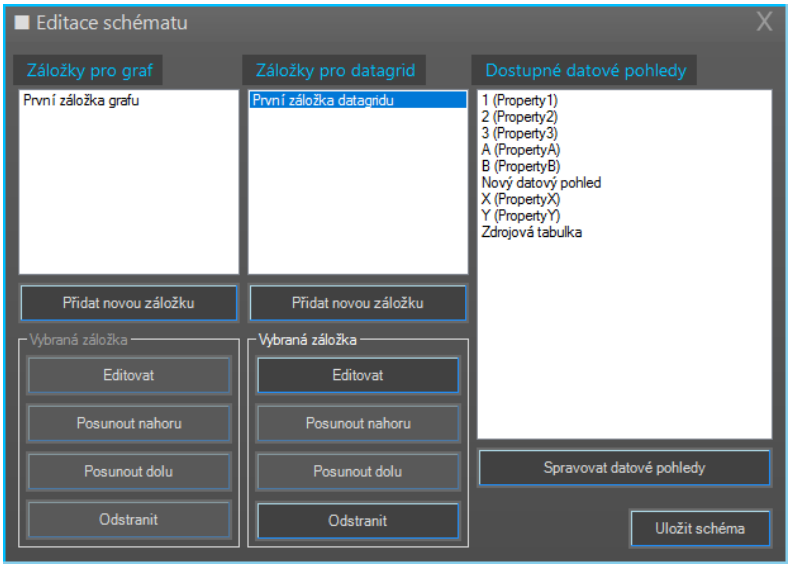
Obrázek 5.14: Editace záložky grafu prvního příkladu



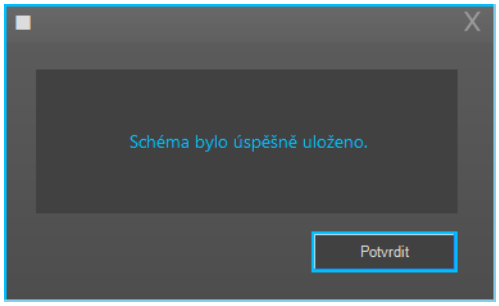
Obrázek 5.15: Editace záložky datagridu prvního příkladu



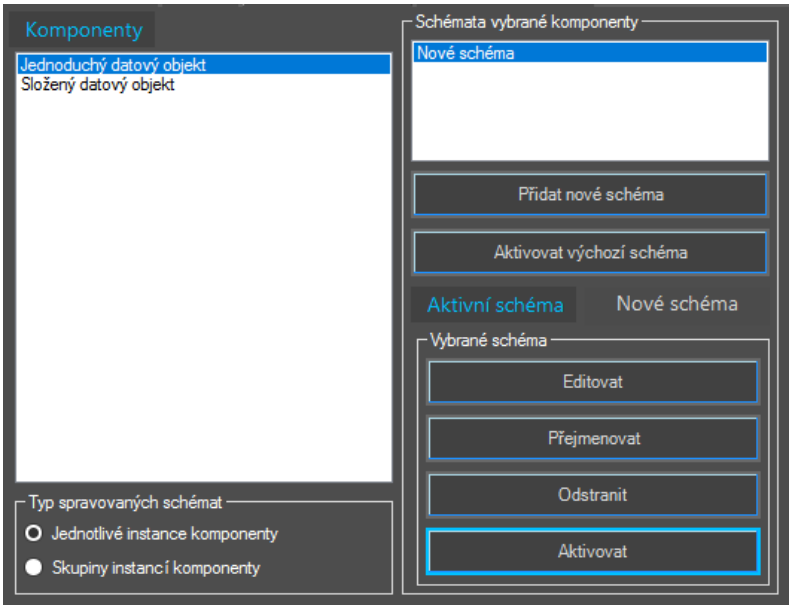
Obrázek 5.16: Výběr pohledů do datagridu prvního příkladu



Obrázek 5.17: Nastavené schéma prvního příkladu



Obrázek 5.18: Úspěšné uložení schématu prvního příkladu

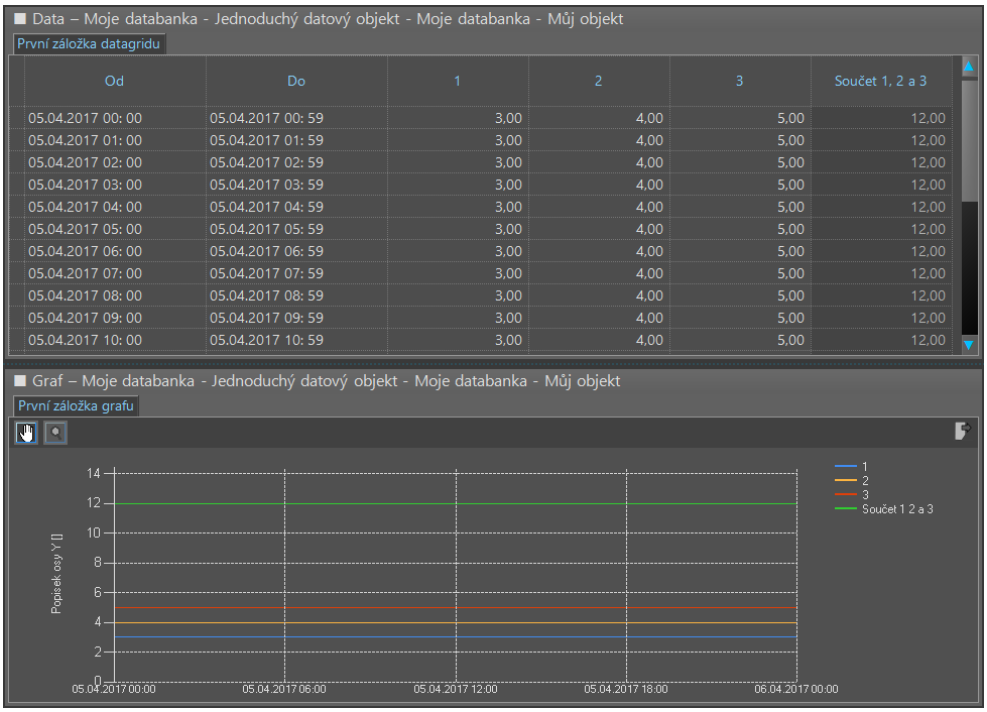


Obrázek 5.19: Aktivované nové schéma prvního příkladu

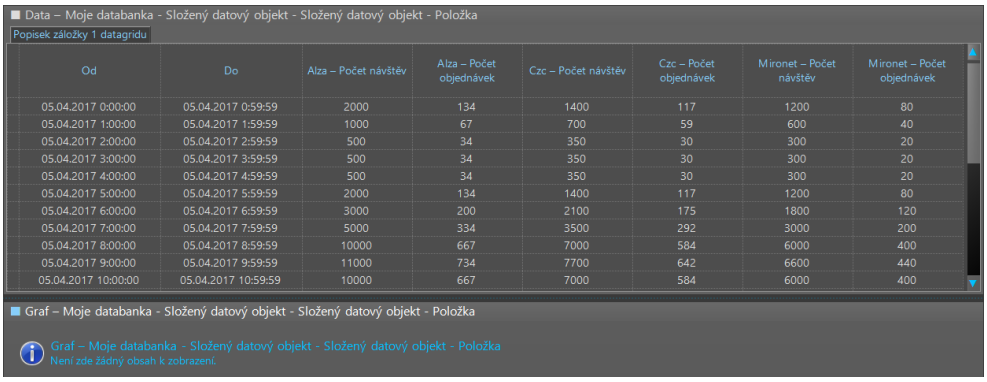
tagridu. Jedná se o stejný příklad jako u všech zkoumaných nástrojů – viz podkapitola 4.1.13 u nástroje SSRS. Ve zmíněné podkapitole se nachází popis a vysvětlení příkladu.

Aby byl příklad méně abstraktní, byl převeden z abstraktní domény (Objekt 1 – Vlastnost 1 apod.) na doménu e-shopů (viz obr. 5.21). Ve zdrojové tabulce jsou smyšlená data pro tři e-shopy – Alza, Czc a Mironet. Pro každý z těchto e-shopů jsou zde dvě časové řady (vlastnosti) – počet návštěv v danou hodinu a počet objednávek v danou hodinu. To, co tedy v původní doméně příkladu byly objekty, jsou nyní e-shopy a to, co byla vlastnost 1 a 2, jsou nyní počet návštěv a objednávek.

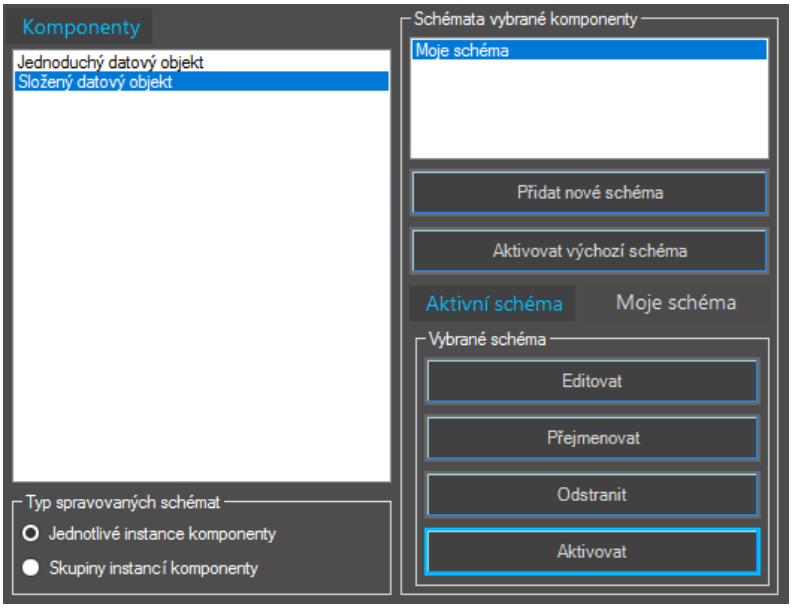
Cílem příkladu je, přidat k těmto sloupcům také součet počtu návštěv a součet počtu objednávek. Dále seřadit řady v pořadí těchto vlastností, aby



Obrázek 5.20: Zobrazení vytvořeného schématu prvního příkladu



Obrázek 5.21: Zobrazení výchozího schématu druhého příkladu



Obrázek 5.22: Správa schémat druhého příkladu

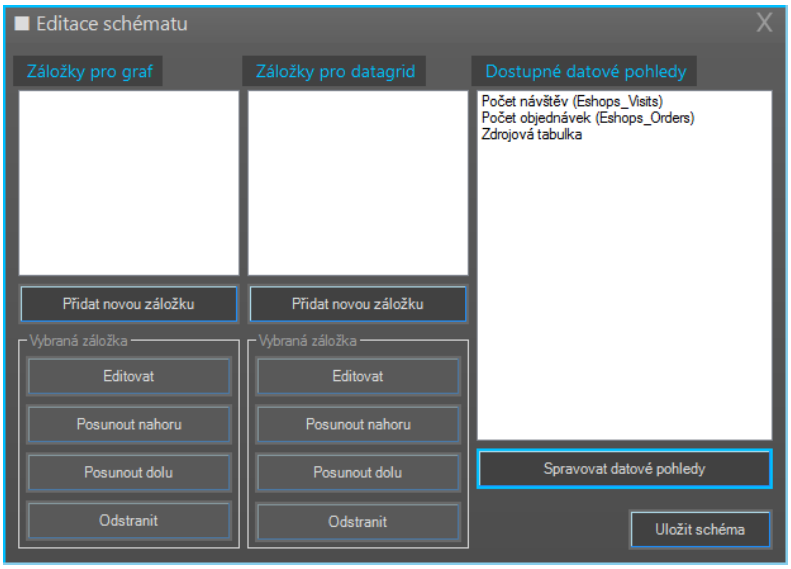
byly u sebe sloupce související s danou vlastností – tedy sloupce s počty návštěv jednotlivých e-shopů následovaných jejich součtem a poté sloupce s počty objednávek jednotlivých e-shopů následované jejich součtem. Nakonec pak zobrazit v datagridu tyto časové řady v rozbalovacích skupinách dle těchto vlastností – v jedné skupince tedy budou časové řady počtů návštěv a v druhé skupince budou časové řady počtu objednávek.

Pro tento příklad byla vytvořena datová třída „Složený datový objekt“, která obsahuje kolekci datových objektů e-shopů, kde datová třída e-shop má časové řady počtu návštěv a počtu objednávek jako dvě properties.

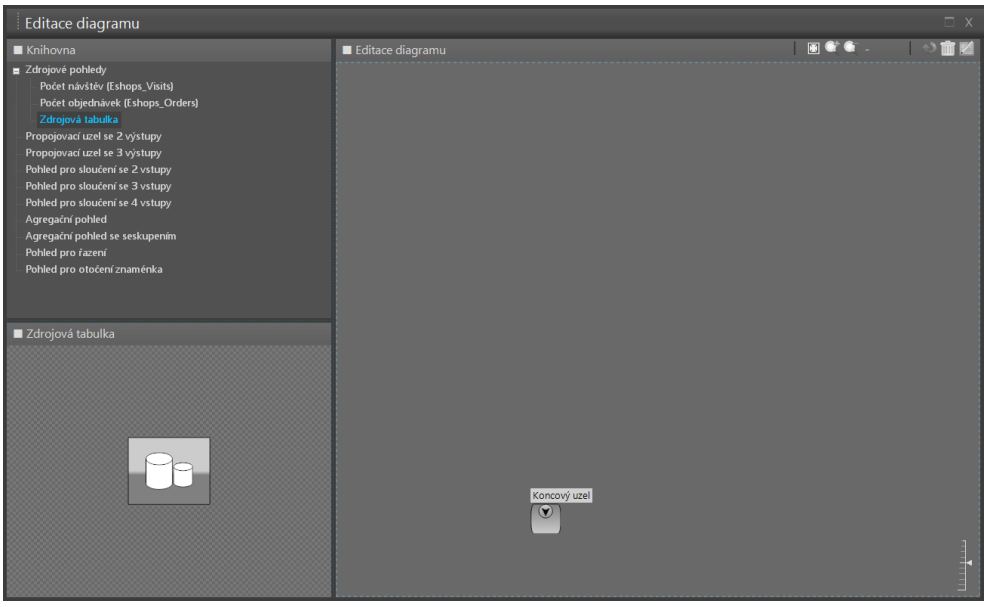
Zobrazení dat pro výchozí schéma tedy opět zobrazuje všechny sloupce v datagridu (viz obr. 5.21). Přesunu se tedy do správy schémat, kde pro datovou třídu založím nové schéma a rovnou ho nastavím jako aktivní (viz obr. 5.22).

Při přechodu na jeho editaci (viz obr. 5.23) vidím, že v dostupných datových pohledech jsou pohledy pro obě vlastnosti. Pod každou touto vlastností se skrývají vždy tři časové řady jednotlivých e-shopů. Nejsou zde jednotlivé časové řady, neboť e-shopy jsou v kolekci, kde jich může být libovolné, dopředu nespecifikované množství. Je tedy potřeba s nimi pracovat na obecnější úrovni, aby schéma nezáviselo na tom, kolik a jaké e-shopy „Složený datový objekt“ v kolekci zrovna obsahuje.

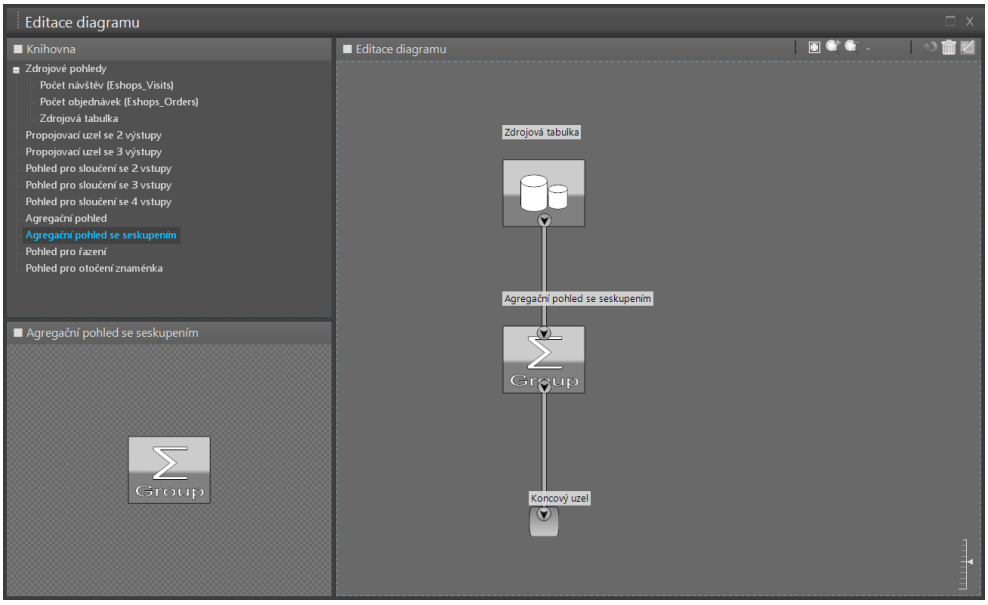
Přesunu se tedy do správy datových pohledů, kde si založím nový strom datových pohledů a přesunu se do jeho editace (viz obr. 5.24), stejně jako v prvním příkladu.



Obrázek 5.23: Editace nového schématu druhého příkladu



Obrázek 5.24: Diagram druhého příkladu – prázdný strom



Obrázek 5.25: Diagram druhého příkladu – agregace se seskupením

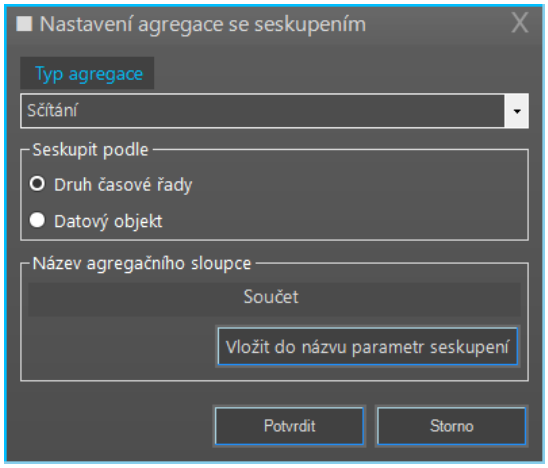
Nejprve si chci připravit součty počtů návštěv a počtů objednávek, tedy součty podle typu vlastnosti. Zdrojem dat bude zdrojová tabulka, kde mám všechny sloupce a dále přidám agregační pohled se seskupením a propojím je (viz obr. 5.25).

Agregační pohled je potřeba nastavit. Otevřu tedy nastavení (viz obr. 5.26) přes jeho kontextové menu (jako u obyčejného agregačního pohledu v prvním příkladu). Skupiny, nad kterými bude agregace provedena, jsou dány dle druhu (vlastnosti) časové řady – tzn., vzniknou dvě skupinky, pro časové řady počtu návštěv a pro časové řady počtu objednávek. Výstupem budou tedy dva součtové sloupce.

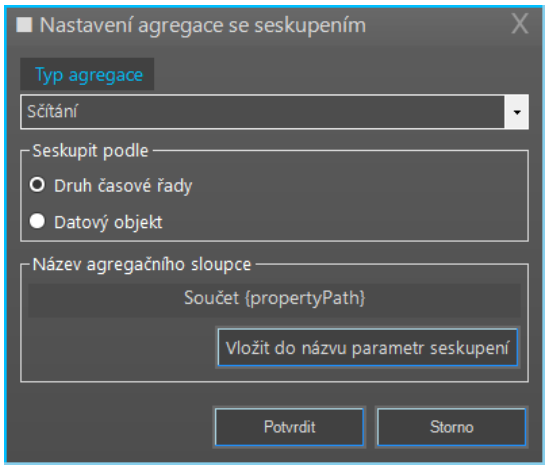
Zbývá tyto sloupce pojmenovat. Nedává smysl pojmenovat každý zvlášť, protože druhů časových řad může být v různých datových třídách různé množství. Proto je zde předpis pro název všech výsledných sloupců (viz obr. 5.27). Tento předpis kromě konstantního řetězce může obsahovat také zástupný symbol pro parametr, podle kterého probíhá seskupení. Tento zástupný symbol mohu vložit tlačítkem „Vložit do názvu parametr seskupení“.

Předpis tedy vypadá následovně: „Součet {propertyPath}“ a výsledné sloupce by tedy měly být pojmenovány jako „Součet Počet návštev“ a „Součet Počet objednávek“. Abych tuto hypotézu ověřil, mohu si zobrazit náhled na data na tento datový pohled. Na obr. 5.28 je vidět, že to tak skutečně je.

Abych zobrazil původní časové řady spolu s těmito součty, přidám do stromu pohled pro sloučení a propojovací uzel (viz obr. 5.29), podobně jako v prvním příkladu.



Obrázek 5.26: Diagram druhého příkladu – nastavení agregace se seskupením

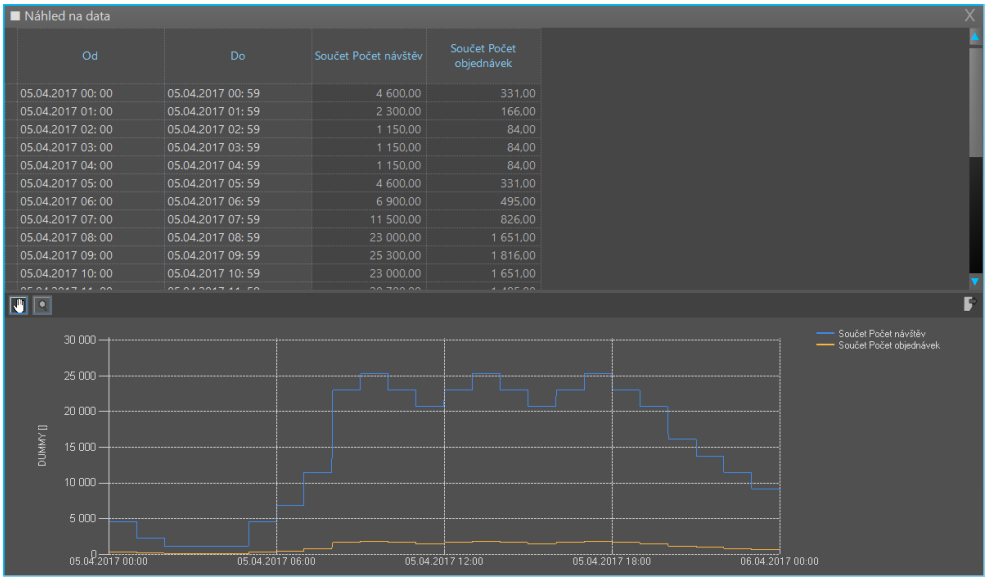


Obrázek 5.27: Diagram druhého příkladu – pojmenování agregačních sloupců

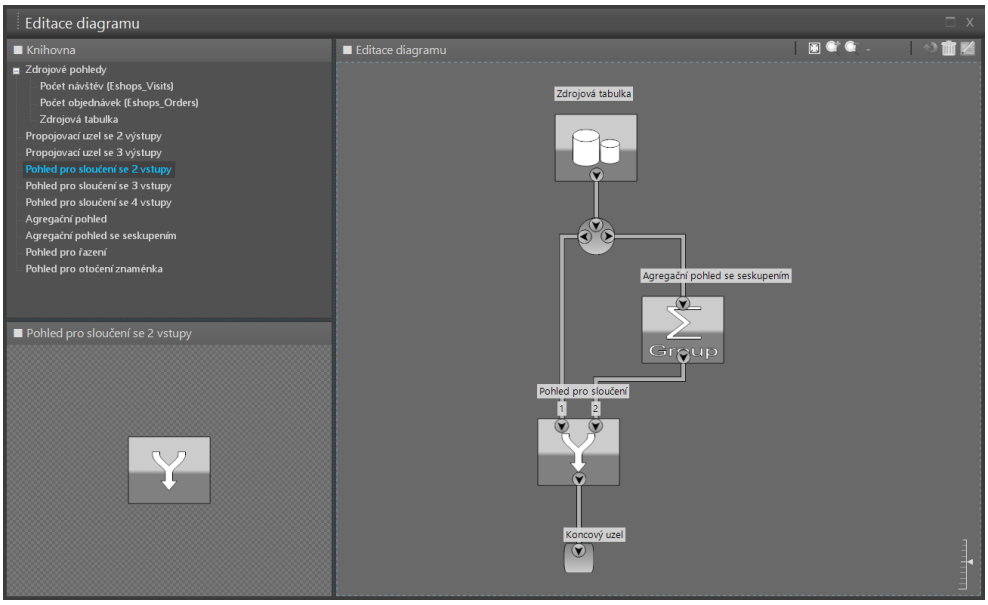
V náhledu na data však vidím (viz obr. 5.30), že sloupce jsou v nesprávném pořadí. Jsou seřazeny dle e-shopů a na konci jsou oba součtové sloupce. Já však chci sloupce seřadit podle jednotlivých vlastností. Tedy počet návštěv jednotlivých e-shopů, jejich součet a dále obdobně pro počet objednávek. K tomu použiji pohled pro řazení, který tedy začlením do stromu (viz obr. 5.31).

Stejně jako u agregace se seskupením je potřeba pohled pro řazení nastavit (viz obr. 5.32). Zde je nastavení jednodušší, stačí pouze vybrat parametr, dle kterého řazení proběhne. V mém případě je to opět druh časové řady.

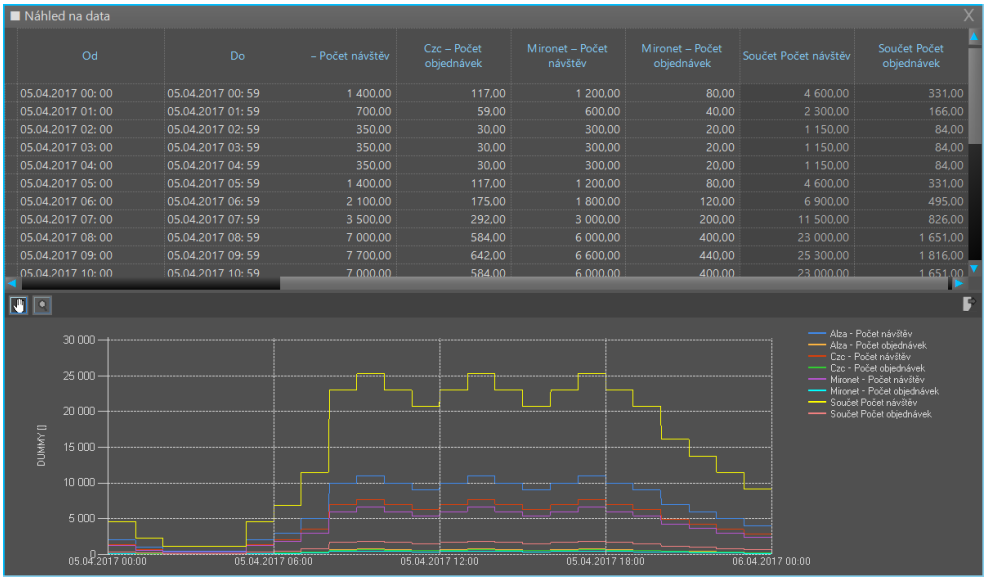
Když si zobrazím náhled na data na tento pohled, (viz obr. 5.33) vidím, že jsou sloupce správně seřazeny (to, že součtové sloupce jsou vždy až po jednotlivých, je dáno pořadím vstupů pohledu pro sloučení).



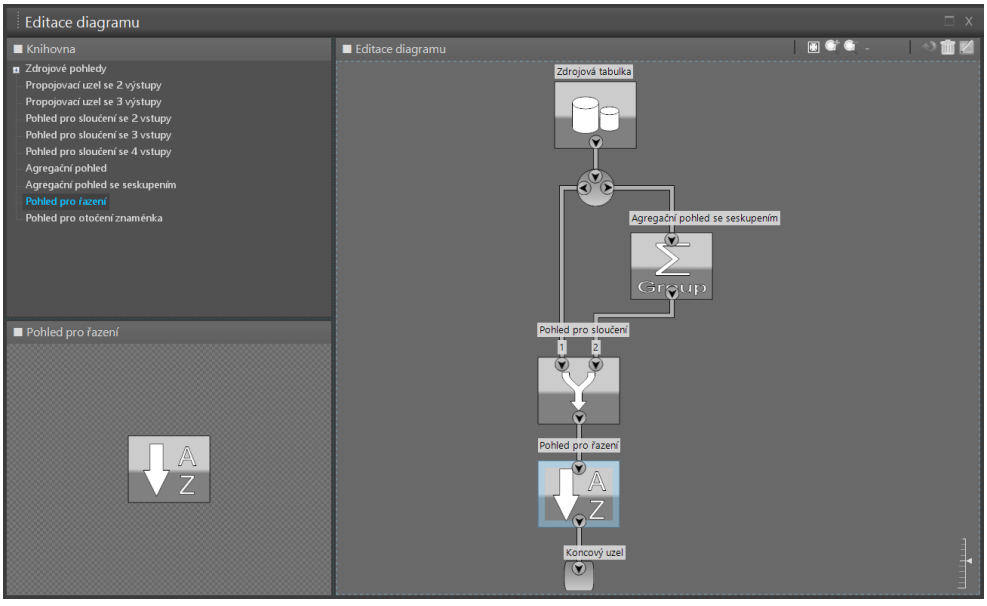
Obrázek 5.28: Diagram druhého příkladu – náhled na data agregace se seskupením



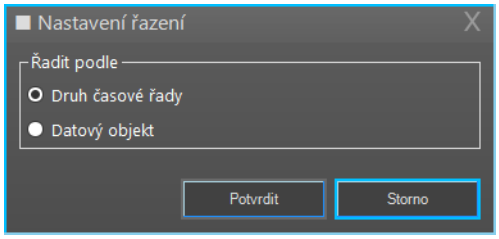
Obrázek 5.29: Diagram druhého příkladu – sloučení agregovaných dat



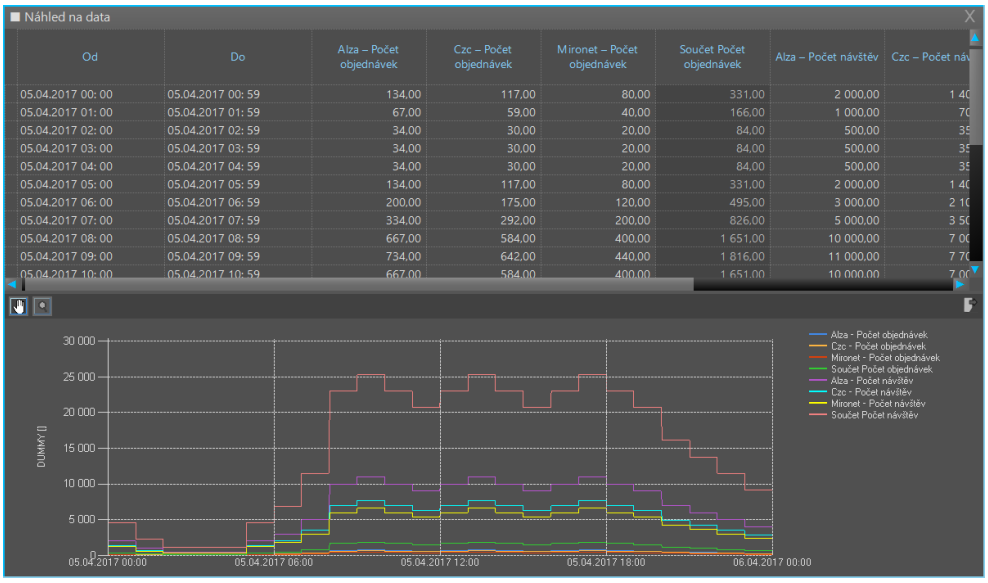
Obrázek 5.30: Diagram druhého příkladu – náhled na neseřazená data



Obrázek 5.31: Diagram druhého příkladu – pohled pro řazení



Obrázek 5.32: Diagram druhého příkladu – nastavení řazení



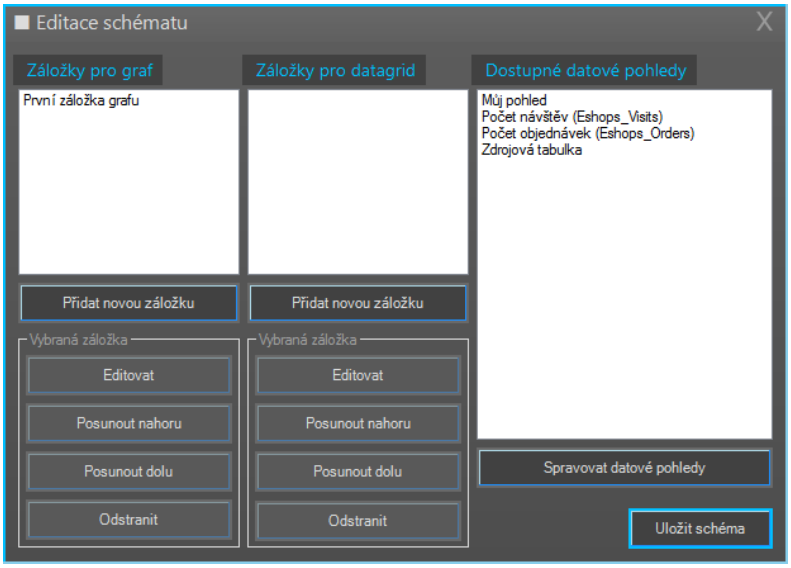
Obrázek 5.33: Diagram druhého příkladu – náhled na seřazená data

Strom je tedy hotov a vracím se zpět do editace schématu (viz obr. 5.34), kde vytvořím jednu záložku pro graf, ve které zobrazím můj vytvořený pohled, stejným způsobem jako u prvního příkladu.

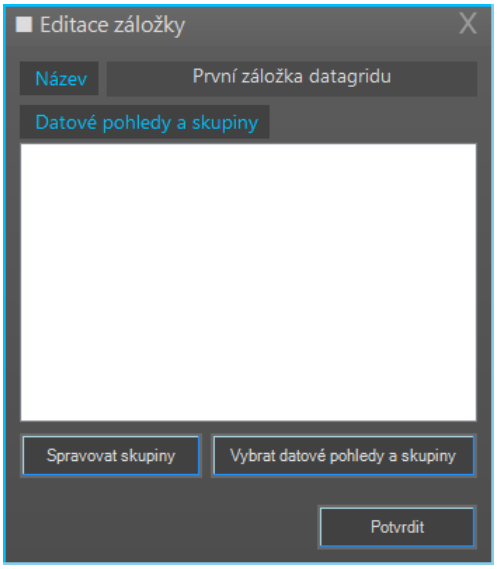
Zbývá vytvořit záložku datagridu – přidám novou záložku a otevřu její editaci (viz obr. 5.35). Pokud bych chtěl jen zobrazit všechny sloupce v této záložce, mohl bych postupovat stejně jako u prvního příkladu.

V zadání příkladu je však požadováno, ale byly v datagridu utvořeny skupinky sloupců dle dané vlastnosti. Přesunu se tedy tlačítkem „Spravovat skupiny“ do správy skupin sloupců (viz obr. 5.36). Mohu zde tvořit buď jednotlivou skupinu sloupců, do které vyberu jednotlivé datové pohledy nebo vytvořit dynamický počet skupin daných parametrem seskupení, podobně jako se tvoří skupiny, nad kterými je provedena agregace v agregačním pohledu se seskupením. V mém případě chci tento způsob, pomocí tlačítka „Přidat skupinu se seskupením“ ji tedy přidám.

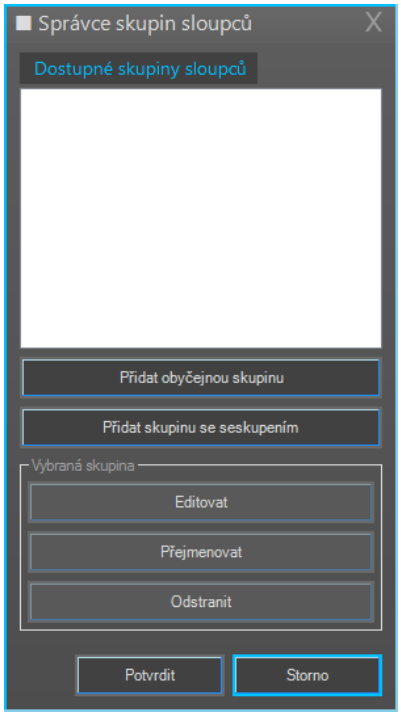
Tím se skupina dostává mezi dostupné skupiny (viz obr. 5.37), je však



Obrázek 5.34: Schéma s novým pohledem a grafem v druhém příkladu



Obrázek 5.35: Editace záložky datagridu v druhém příkladu



Obrázek 5.36: Správce skupin sloupců v datagridu

potřeba ji nastavit. Přejdu tedy do nastavení (viz obr. 5.38) pomocí tlačítka „Editovat“. Nejprve je potřeba vybrat, jaké datové pohledy budou ve skupině (horní část okna). Pro to slouží stejné okno pro výběr, se kterým jsme se již setkali (viz obr. 5.16) – pomocí něj tedy vyberu můj datový pohled. Dále ve spodní části okna určím parametr seskupení, kterým bude opět druh časové řady, neboť chci mít skupinu sloupců pro každou vlastnost. Tím je skupina připravena, vrátím se tedy k editaci záložky.

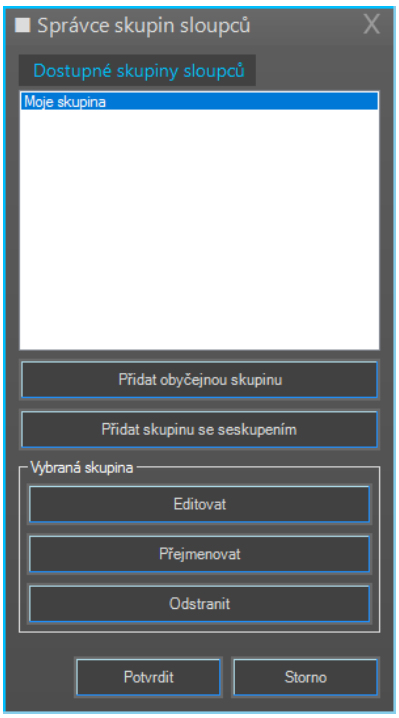
Zde se viditelně nic nezměnilo (viz obr. 5.39), jsou již připravené skupiny sloupců. Abych je mohl zobrazit v datagridu, otevřu tedy „Vybrat datové pohledy a skupiny“.

Jak je vidět, mám zde kromě datových pohledů k dispozici i nově vytvořenou skupiny a tedy ji přidám (viz obr. 5.40).

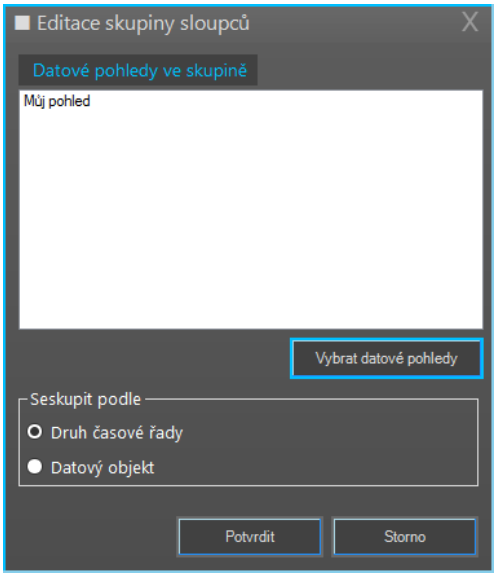
Tím je záložka nastavena a celé schéma je tedy kompletní. Když teď zobrazím v aplikaci data k této datové třídě, data jsou již v požadované podobě (viz obr. 5.41 – první skupina sloupců je sbalená, druhá je rozbalená; po sbalení je zobrazen pouze nejpravější sloupec, v mém případě tedy součtový).

5.1.3 Rozbor předpisu prvního příkladu

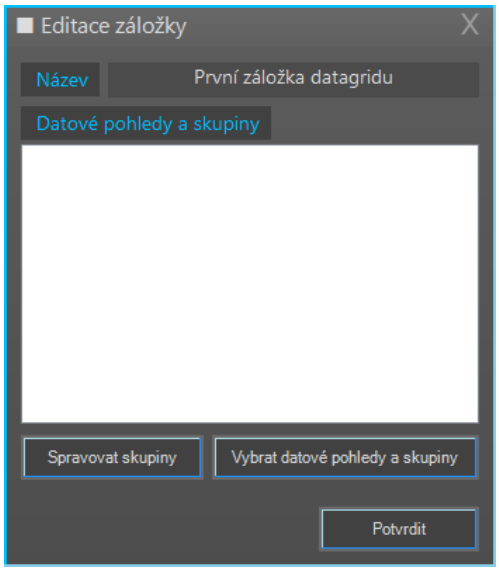
Podobně jako u zkoumaného nástroje SSRS, u implementovaného nástroje nahlédnu do vygenerovaných Předpisů, které vznikly při uložením schémat,



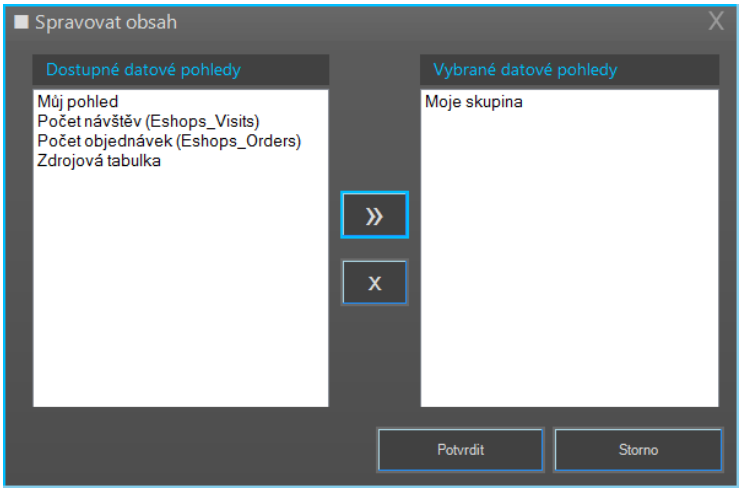
Obrázek 5.37: Správce skupin sloupců s novou skupinou



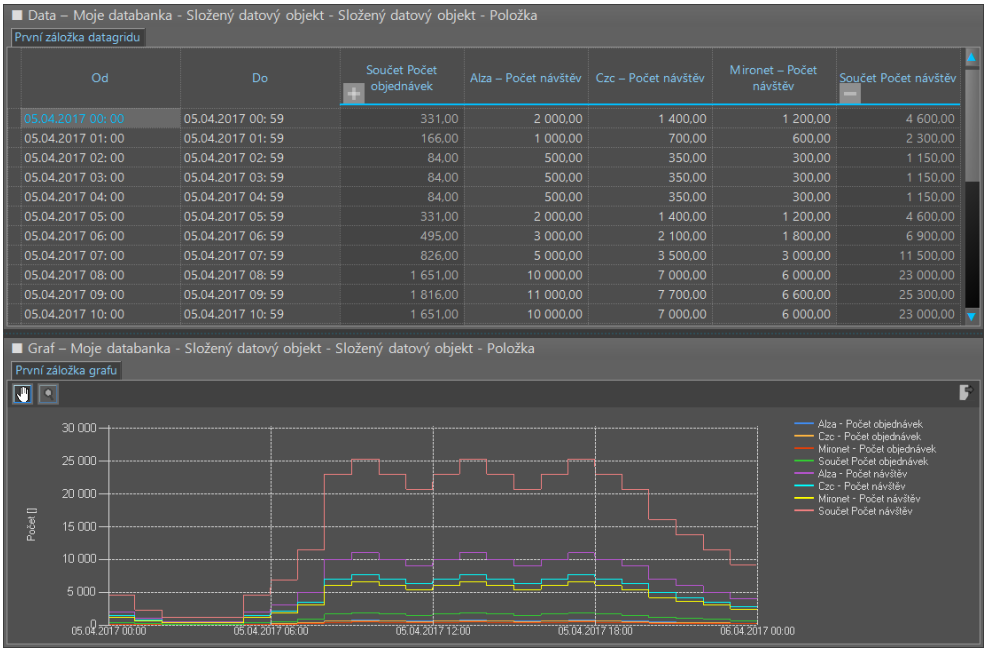
Obrázek 5.38: Editace skupiny sloupců se seskupením



Obrázek 5.39: Dokončení editace záložky datagridu druhého příkladu



Obrázek 5.40: Přidání vytvořené skupiny sloupců do datagridu



Obrázek 5.41: Zobrazení vytvořeného schématu druhého příkladu

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <scheme name="Moje schéma">
3     <tables>...</tables>
6     <trees>...</trees>
63    <chartViews>...</chartViews>
66    <datagridViews>...</datagridViews>
69    <columnSelections>...</columnSelections>
103   <chartStyles>...</chartStyles>
109   <datagridStyles>...</datagridStyles>
116   <datagridColumnStyles>...</datagridColumnStyles>
123   </scheme>
```

Obrázek 5.42: Náhled na Předpis prvního příkladu

```
3   <tables>
4     <table tableId="zaklad" viewId="zaklad" />
5   </tables>
```

Obrázek 5.43: Sekce Tables Předpisu prvního příkladu

vytvořených v předchozí podkapitole. Oba celé tyto Předpisy se nachází na přiloženém DVD.

V této podkapitole se podívám na Předpis, který vznikl ze schématu prvního příkladu.

Předpis má 123 řádek a je členěn do sekcí, jejichž význam byl podrobněji popsán v kapitole 3.3 (viz obr. 5.42).

V sekci **tables** (viz obr. 5.43), se nachází definice jediné zdrojové tabulky

```
<trees>
  <tree treeId="Můj pohled Pohled pro sloučení">
    <merge viewId="Můj pohled Pohled pro sloučení" overwriteParentViewName="True">
      <subtree treeId="Property1" />
      <subtree treeId="Property2" />
      <subtree treeId="Property3" />
    </merge>
  </tree>
  <tree treeId="Property1">
    <selection viewId="Property1" columnSelectionIds="Filter  Property1">
      <table tableId="zaklad" />
    </selection>
  </tree>
  <tree treeId="Property2">...</tree>
  <tree treeId="Property3">...</tree>
  <tree treeId="PropertyA">...</tree>
  <tree treeId="PropertyB">...</tree>
  <tree treeId="Můj pohled">
    <merge viewId="Můj pohled" overwriteParentViewName="True">
      <subtree treeId="Můj pohled Pohled pro sloučení" />
      <aggregation viewId="Můj pohled Agregační pohled" aggregationId="Můj pohled Agregační pohled" operation="addition"
                   columnNameBlueprintResource=
                   "SimpleDataObject_SingleInstance_MojeSchema_NameBlueprintOfAggregationColumn_MujPohledAgregacniPohled">
        <subtree treeId="Můj pohled Pohled pro sloučení" />
      </aggregation>
    </merge>
  </tree>
  <tree treeId="PropertyX">...</tree>
  <tree treeId="PropertyY">...</tree>
  <tree treeId="První záložka datagridu">
    <merge viewId="První záložka datagridu" overwriteParentViewName="False">
      <subtree treeId="Můj pohled" />
    </merge>
  </tree>
</trees>
```

Obrázek 5.44: Sekce Trees Předpisu prvního příkladu

schématu.

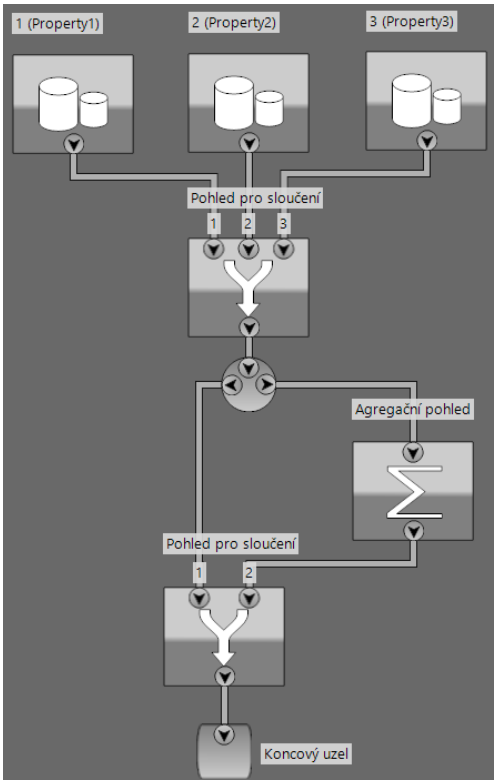
V sekci **trees** (viz obr. 5.44) se nachází všechny stromy datových pohledů.

Strom „Property1“ (element **tree** s touto hodnotou v atributu **treeId**) odpovídá datovému pohledu, který označuje vlastnost 1 a byl již součástí základní sady datových pohledů. Strom obsahuje pouze datový pohled pro výběr, který je aplikován na zdrojovou tabulku a vybírá z ní odpovídající časovou řadu. Stejným způsobem jsou zde pak stromy pro vlastnosti 2, 3, A, B, X a Y – jejich obsah je identický, a proto jsou na obrázku pro větší přehlednost skryty.

Strom „Můj pohled Pohled pro sloučení“ odpovídá v diagramu (viz obr. 5.45) podstromu, jehož kořen je horní pohled pro sloučení a listy jsou pohledy vlastností 1, 2 a 3. Je vidět, že i v Předpisu je datový pohled pro sloučení (element **merge**), který slučuje datové pohledy (podstromy) jednotlivých vlastností.

Zbytek diagramu je pak popsán stromem „Můj pohled“, jehož kořen je spodní pohled pro sloučení – tedy výsledný pohled diagramu a list je pak podstrom „Můj pohled Pohled pro sloučení“. Struktura datových pohledů v diagramu opět odpovídá struktuře v Předpisu. Je zde vidět, že pohled pro sloučení má dva potomky – zmíněný list a agregační pohled, a že agregační pohled je nad tímto listem.

Zbývá už jen strom „První záložka datagridu“, který definuje, co je zobrazeno v záložce datagridu. V Předpise má datagrid vždy jen jeden zdrojový pohled a aby bylo možné jich mít v datagridu více, tak jako je to umožněno



Obrázek 5.45: Strom datových pohledů prvního příkladu

zadat v GUI, tak je zde tento strom, který by tyto vybrané pohledy obalil pohledem pro sloučení a zajistil tak, že datagrid bude mít v Předpisu na vstupu opět jen jeden pohled. V tomto případě, je ale tento mechanismus nadbytečný, vzhledem k tomu, že byl v GUI vybrán jen jeden datový pohled.

V sekci **columnSelections** (viz obr. 5.46) jsou popsány selekce sloupců použité v datových pohledech ve stromech. Jsou zde selekce pro všechny vlastnosti datových objektů a dále selekce pro všechny datové pohledy (které byly vytvořeny ve stromech).

V sekcích **chartViews** a **chartStyles** (viz obr. 5.47) je popsáno nastavení grafu a jeho záložek. Je zde definován popisek osy Y (element **axisYTitleResource**). Dále je zde popsána jedna záložka, která se nachází v grafu (element **view**), je u ní odkaz na datový pohled (element **viewId**) a dále je zde popisek záložky (element **chartTabLabelResource**).

Zbývá popis sekcí datagridu (viz obr. 5.48) – elementy **datagridViews**, **datagridStyles** a **datagridColumnStyles**. V sekci datagridViews je vidět, že datagrid má jednu záložku a také je zde vidět její navázání na datový pohled. Styly datagridu v tomto případě nejsou příliš zajímavé na rozbor.

```
<columnSelections>
  <columnSelection columnSelectionId="Property1" type="parentViewName" parentViewName="Property1" />
  <columnSelection columnSelectionId="Filter  Property1" type="normal">
    <propertyPath>Property1</propertyPath>
  </columnSelection>
  <columnSelection columnSelectionId="Property2" type="parentViewName" parentViewName="Property2" />
  <columnSelection columnSelectionId="Filter  Property2" type="normal">
    <propertyPath>Property2</propertyPath>
  </columnSelection>
  <columnSelection columnSelectionId="Property3" type="parentViewName" parentViewName="Property3" />
  <columnSelection columnSelectionId="Filter  Property3" type="normal">
    <propertyPath>Property3</propertyPath>
  </columnSelection>
  <columnSelection columnSelectionId="PropertyA" type="parentViewName" parentViewName="PropertyA" />
  <columnSelection columnSelectionId="Filter  PropertyA" type="normal">
    <propertyPath>PropertyA</propertyPath>
  </columnSelection>
  <columnSelection columnSelectionId="PropertyB" type="parentViewName" parentViewName="PropertyB" />
  <columnSelection columnSelectionId="Filter  PropertyB" type="normal">
    <propertyPath>PropertyB</propertyPath>
  </columnSelection>
  <columnSelection columnSelectionId="Můj pohled Pohled pro sloučení" type="parentViewName"
                   parentViewName="Můj pohled Pohled pro sloučení" />
  <columnSelection columnSelectionId="Můj pohled Agregační pohled" type="parentViewName"
                   parentViewName="Můj pohled Agregační pohled" />
  <columnSelection columnSelectionId="Můj pohled" type="parentViewName" parentViewName="Můj pohled" />
  <columnSelection columnSelectionId="PropertyX" type="parentViewName" parentViewName="PropertyX" />
  <columnSelection columnSelectionId="Filter  PropertyX" type="normal">
    <propertyPath>PropertyX</propertyPath>
  </columnSelection>
  <columnSelection columnSelectionId="PropertyY" type="parentViewName" parentViewName="PropertyY" />
  <columnSelection columnSelectionId="Filter  PropertyY" type="normal">
    <propertyPath>PropertyY</propertyPath>
  </columnSelection>
  <columnSelection columnSelectionId="zaklad" type="parentViewName" parentViewName="zaklad" />
</columnSelections>
```

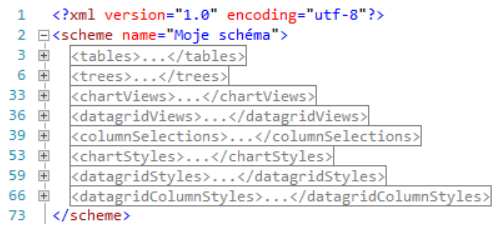
Obrázek 5.46: Sekce ColumnSelections Předpisu prvního příkladu

```
<chartViews>
  <view viewId="Můj pohled" chartStyleId="První záložka grafu"
        chartTabLabelResource="SimpleDataObject_SingleInstance_MojeSchema_LabelOfChartTab_PrvniZalozkaGrafu" />
</chartViews>
<datagridViews>...</datagridViews>
<columnSelections>...</columnSelections>
<chartStyles>
  <chartStyle chartStyleId="První záložka grafu">
    <axisYTitleResource>SimpleDataObject_SingleInstance_MojeSchema_AxisYLabelOfTab_PrvniZalozkaGrafu</axisYTitleResource>
  </chartStyle>
  <defaultChartStyle />
</chartStyles>
```

Obrázek 5.47: Sekce grafu Předpisu prvního příkladu

```
<datagridViews>
  <view viewId="První záložka datagridu" datagridStyleId="První záložka datagridu"
        datagridTabLabelResource="SimpleDataObject_SingleInstance_MojeSchema_LabelOfDatagridTab_PrvniZalozkaDatagridu" />
</datagridViews>
<columnSelections>...</columnSelections>
<chartStyles>...</chartStyles>
<datagridStyles>
  <datagridStyle datagridStyleId="První záložka datagridu" anchorRowNumber="0" anchorTopLevelSelectionNumber="0">
    <datagridColumnSelection indexColumnStyleId="INDEX_COLUMN_STYLE">
      <columnSelection columnSelectionId="Můj pohled" />
    </datagridColumnSelection>
  </datagridStyle>
</datagridStyles>
<datagridColumnStyles>
  <datagridColumnStyle datagridColumnStyleId="INDEX_COLUMN_STYLE">
    <formatResource>SimpleDataObject_SingleInstance_MojeSchema_IndexColumnFormat</formatResource>
    <alignment>middleLeft</alignment>
  </datagridColumnStyle>
  <defaultDatagridColumnStyle />
</datagridColumnStyles>
```

Obrázek 5.48: Sekce datagridu Předpisu prvního příkladu



Obrázek 5.49: Náhled na Předpis druhého příkladu

```
<trees>
  <tree treeId="Můj pohled">
    <orderBy viewId="Můj pohled" orderingCriteria="propertyPath">
      <merge viewId="Můj pohled Pohled pro sloučení" overwriteParentViewName="True">
        <table tableId="zaklad" />
        <groupByAggregation viewId="Můj pohled Agregační pohled se seskupením"
                            aggregationId="Můj pohled Agregační pohled se seskupením"
                            operation="addition"
                            columnNameBlueprintResource="ComposedDataObject...MujPohledAgregacniPohledSeSeskupenim"
                            groupingParameter="propertyPath">
          <table tableId="zaklad" />
        </groupByAggregation>
      </merge>
    </orderBy>
  </tree>
  <tree treeId="Eshops_Visits">
    <selection viewId="Eshops_Visits" columnSelectionIds="Filter  Eshops_Visits">
      <table tableId="zaklad" />
    </selection>
  </tree>
  <tree treeId="Eshops_Orders">
    <selection viewId="Eshops_Orders" columnSelectionIds="Filter  Eshops_Orders">
      <table tableId="zaklad" />
    </selection>
  </tree>
  <tree treeId="První záložka datagridu">
    <merge viewId="První záložka datagridu" overwriteParentViewName="False">
      <subtree treeId="Můj pohled" />
    </merge>
  </tree>
</trees>
```

Obrázek 5.50: Sekce Trees Předpisu druhého příkladu

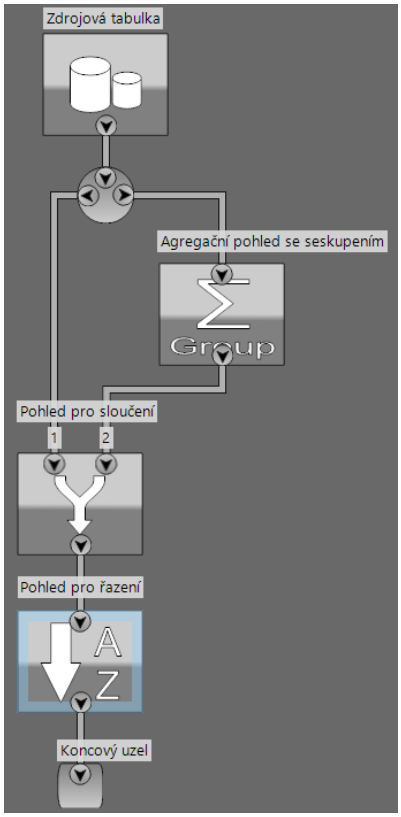
5.1.4 Rozbor předpisu druhého příkladu

Podobným způsoben bude rozebrán i Předpis druhého vytvořeného schématu (viz obr. 5.49).

Sekce **tables** je identická jako u prvního příkladu.

Sekce **trees** (viz obr. 5.50) obsahuje dva stromy pro datové pohledy označující jednotlivé vlastnosti – počty návštěv a počty objednávek. Dále je zde strom popisující záložku datagridu, podobně jako u prvního příkladu.

Nakonec tu je strom reprezentující diagram datových pohledů (viz obr. 5.51) „Můj pohled“. Strom v Předpise opisuje strom v diagramu. Je vidět, že je při řazení a seskupování (elementy **orderBy** a **groupByAggregation**) použito kritérium druh časové řady, resp. **propertyPath**. Na rozdíl od prvního příkladu je zde pro popis diagramu pouze jeden strom a nikoli více. Je to tím, že tento graf v tomto diagramu bylo možné mapovat na jeden strom v Předpise, zatímco u prvního příkladu nikoli a tak musel být graf dekomponován na dva stromy. Algoritmus pro dekompozici grafu pohledů v diagramu



Obrázek 5.51: Strom datových pohledů druhého příkladu

```
<columnSelections>
  <columnSelection columnSelectionId="Můj pohled Agregační pohled se seskupením" type="parentViewName"
                   parentViewName="Můj pohled Agregační pohled se seskupením" />
  <columnSelection columnSelectionId="Můj pohled Pohled pro sloučení" type="parentViewName"
                   parentViewName="Můj pohled Pohled pro sloučení" />
  <columnSelection columnSelectionId="Můj pohled" type="parentViewName" parentViewName="Můj pohled" />
  <columnSelection columnSelectionId="Eshops_Visits" type="parentViewName" parentViewName="Eshops_Visits" />
  <columnSelection columnSelectionId="Filter  Eshops_Visits" type="normal">
    <propertyPath>Eshops_Visits</propertyPath>
  </columnSelection>
  <columnSelection columnSelectionId="Eshops_Orders" type="parentViewName" parentViewName="Eshops_Orders" />
  <columnSelection columnSelectionId="Filter  Eshops_Orders" type="normal">
    <propertyPath>Eshops_Orders</propertyPath>
  </columnSelection>
  <columnSelection columnSelectionId="zaklad" type="parentViewName" parentViewName="zaklad" />
</columnSelections>
```

Obrázek 5.52: Sekce ColumnSelections Předpisu druhého příkladu

bude popsán později.

Sekce **columnSelections** vypadá (viz obr. 5.52) stejným způsobem jako u prvního příkladu, jsou zde tedy selekce pro všechny datové pohledy a pro výběr jednotlivých properties datové třídy.

Sekce **chartViews** a **chartStyles** jsou obdobné jako u prvního příkladu.

Sekce **datagridViews** a **datagridColumnStyles** jsou vedeny stejným

```
<datagridViews>
  <view viewId="První záložka datagridu" datagridStyleId="První záložka datagridu"
        datagridTabLabelResource="ComposedDataObject_SingleInstance_MojeSchema_LabelOfDatagridTab_PrvniZalozkaDatagridu" />
</datagridViews>
<columnSelections>...</columnSelections>
<chartStyles>...</chartStyles>
<datagridStyles>
  <datagridStyle datagridStyleId="První záložka datagridu" anchorRowNumber="0" anchorTopLevelSelectionNumber="0">
    <datagridColumnSelection indexColumnStyleId="INDEX_COLUMN_STYLE">
      <groupBySelection expandDirection="left" isExpanded="False" groupingParameter="propertyPath" columnSelectionIds="Můj pohled" />
    </datagridColumnSelection>
  </datagridStyle>
</datagridStyles>
<datagridColumnStyles>
  <datagridColumnStyle datagridColumnStyleId="INDEX_COLUMN_STYLE">
    <formatResource>ComposedDataObject_SingleInstance_MojeSchema_IndexColumnFormat</formatResource>
    <alignment>middleLeft</alignment>
  </datagridColumnStyle>
  <defaultDatagridColumnStyle />
</datagridColumnStyles>
```

Obrázek 5.53: Sekce datagridu Předpisu druhého příkladu

způsobem jako v prvním příkladu (viz obr. 5.53). V sekci **datagridStyles** je definice skupin sloupců, které jsou utvořeny opět dle druhu časové řady, tedy **propertyPath**.

5.2 Popis implementace

Aplikace je implementována v jazyce C# a pro uživatelské rozhraní je zvolena knihovna Windows Forms. Nástroj je součástí této aplikace a proto používá rovněž tyto technologie.

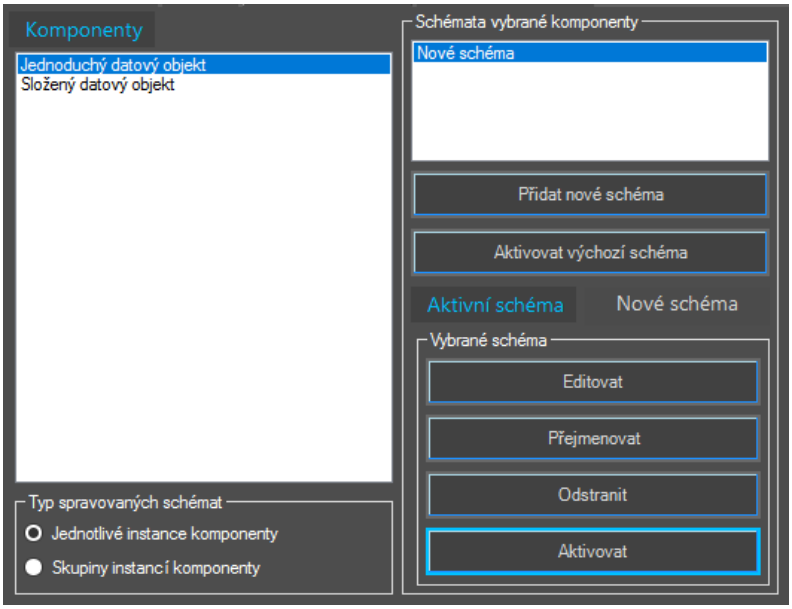
V této kapitole následuje shrnující popis celého modulu, dále rozbor jednotlivých jeho částí a nakonec diskuze implementovaných tříd.

5.2.1 Souhrnný popis nástroje

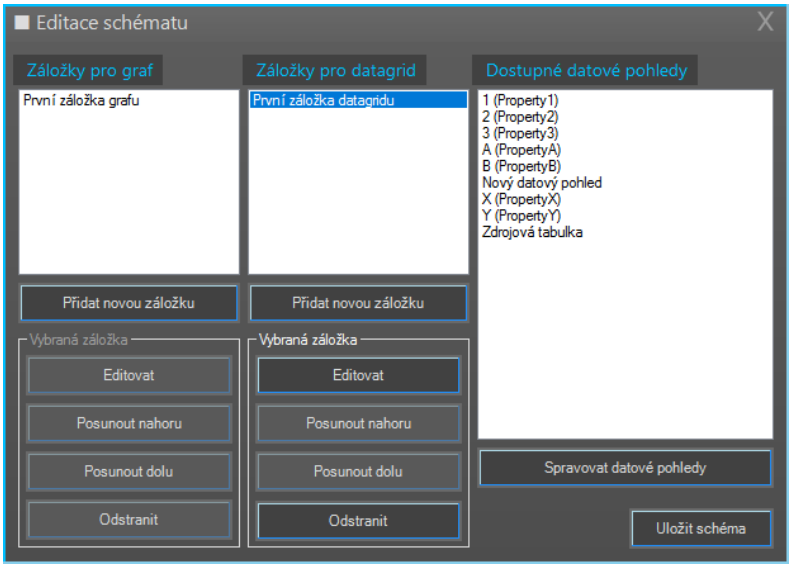
Správa schémat – úvodní obrazovka, umožňuje přidat nová a spravovat existující schémata jednotlivých datových tříd (viz obr. 5.54). Tato schémata popisují, co se v aplikaci pro tyto třídy zobrazí v datagridech a grafech.

Editace vybraného schématu – umožňuje spravovat dostupné datové pohledy schématu a definovat v něm záložky datagridu a grafu (viz obr. 5.55). U **definice záložek** se především určí, jaké datové pohledy se zobrazí v jakých záložkách a jakým způsobem. **Správa datových pohledů** – umožňuje tvorbu datových pohledů a jejich vzájemné propojení do stromových struktur. Tím se zajistí, aby byla zdrojová data zpracována, tedy vyfiltrována, setříděna nebo agregována požadovaným způsobem.

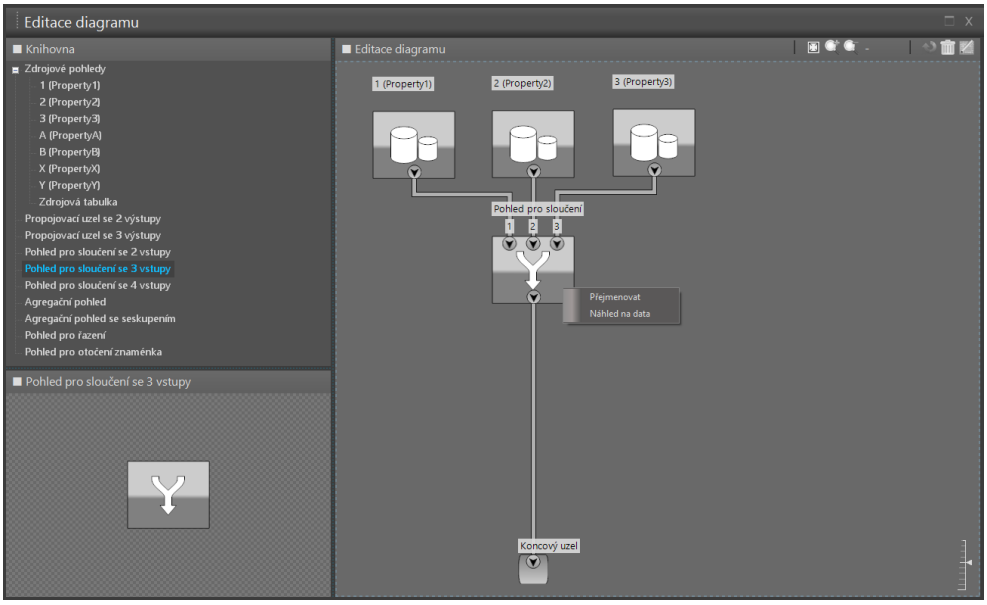
Editace datových pohledů probíhá v diagramu datových pohledů (viz obr. 5.56). V diagramu jsou datové pohledy jako uzly v grafu, se vstupními a výstupními konektory, které lze vzájemně propojit pomocí hran. Takto vytvořený graf popisuje zpracování dat od zdrojových datových pohledů obsahujících surová data až do výsledných datových pohledů s přehlednou podobou dat, které jsou v aplikaci zobrazeny uživateli.



Obrázek 5.54: Správa schemat



Obrázek 5.55: Editace vybraného schématu



Obrázek 5.56: Editace stromu datových pohledů

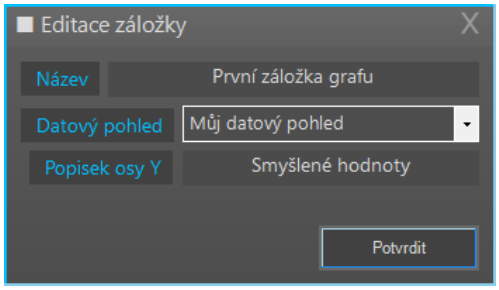
5.2.2 Správa schémat

Správa schémat probíhá v úvodním formuláři celého nástroje (viz obr. 5.54). Je zde seznam datových tříd / komponent a ke každé datové třídě je seznam schémat. Schéma definuje, co se při zobrazení instance této datové třídy zobrazí za data v datagridu a grafu. Z počátku má každá komponenta pouze výchozí schéma, které zobrazuje všechna její data (časové řady) v jedné záložce datagridu (viz dříve na obr. 5.1). Schéma lze v tomto formuláři přidat, odebrat, přejmenovat nebo otevřít formulář pro jeho editaci. Dále lze určit, jaké schéma je právě aktivní – podle něj se k datové třídě v aplikaci budou zobrazovat data. Poslední věcí ve správě schémat je nastavení typu schémat (vlevo dole na obrázku). Datová třída může mít jiné schéma, pokud je zobrazena jedna její instance a jiné schéma pro zobrazení skupiny instancí této třídy. U skupiny instancí se typicky provedou nad daty další agregace jako např. souhrny přes všechny tyto instance apod.

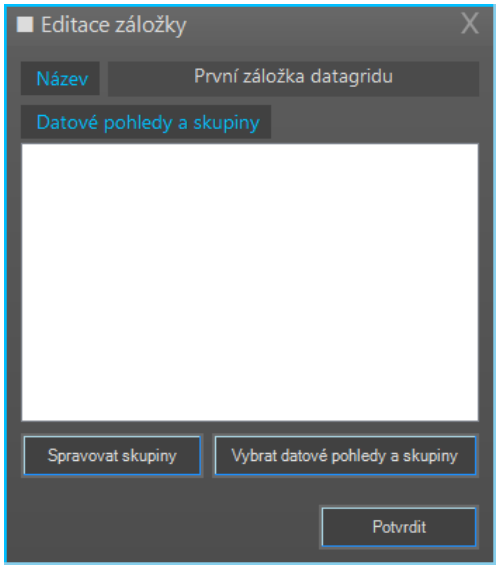
5.2.3 Editace vybraného schématu

Do formuláře editace vybraného schématu (viz obr. 5.55) se lze dostat ze správy schémat. Z tohoto formuláře lze spravovat datové pohledy schématu (viz podkapitola 5.2.5) a jeho záložky pro graf a datagrid (viz podkapitola 5.2.4). V tomto formuláři lze také schéma uložit.

Uložení schématu proběhne serializací objektu obsahující data celého schématu do JSON formátu, ze kterého lze pak celé schéma do nástroje opět načíst.



Obrázek 5.57: Editace záložky grafu



Obrázek 5.58: Editace záložky datagridu

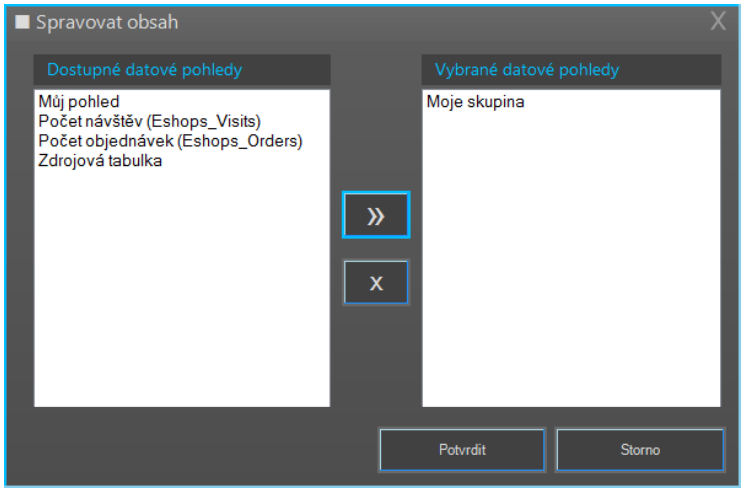
Dále jsou při uložení vygenerovány výstupní soubory, ze kterých je schéma načteno před jeho zobrazením v aplikaci.

Tyto výstupní soubory zahrnují Předpis v Jazyce v souboru ve formátu XML a dále Resource file s lokalizací textových řetězců použitých ve schématu. Resource file umožňuje stejné schéma lokalizovat do několika jazyků – při tvorbě se vygeneruje Resource file s jazykem, ve kterém byl nástroj spuštěn a tento Resource je pak možné editovat (mimo implementovaný nástroj) a přidat do něj lokalizaci v dalších jazycích (např. v angličtině).

5.2.4 Editace záložek grafu a datagridu

Do editace záložek grafu a datagridu se lze dostat z formuláře editace vybraného schématu.

Editace záložky grafu (viz obr. 5.57) je velmi jednoduchá. Je zde zadání ná-



Obrázek 5.59: Výběr datových pohledů na záložku datagridu

zvu záložky, popisku osy Y a dále lze z nabídky dostupných datových pohledů vybrat pohled, který bude v záložce zobrazen.

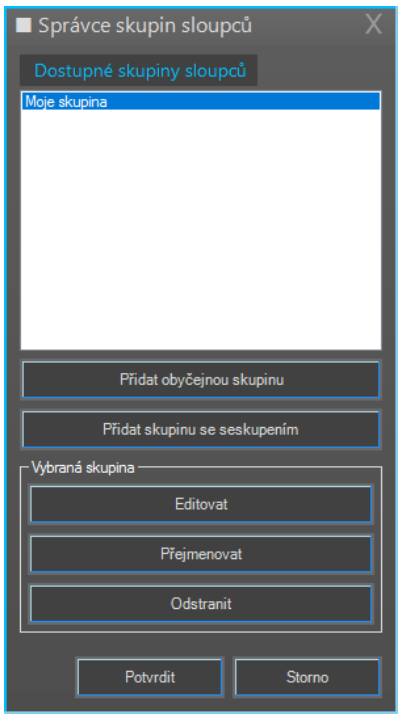
Editace záložky datagridu (viz obr. 5.58) je trochu složitější než u grafu. Je zde zadání názvu záložky, dále zde lze spravovat dostupné skupiny sloupců v záložce (tlačítko „Spravovat skupiny“) a následně vybrat jaké datové pohledy a skupiny sloupců budou v záložce zobrazeny (tlačítko „Vybrat datové pohledy a skupiny“).

Formulář pro výběr jaké pohledy a skupiny sloupců budou použity, pouze umožňuje vybrat podmnožinu prvků z této množiny a určit jim pořadí, v jakém budou zobrazeny (viz obr. 5.59).

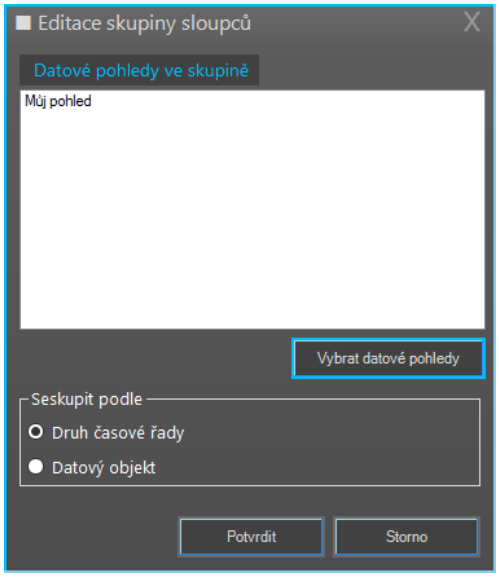
Ve správě skupin sloupců lze skupiny přidávat, otevřít okno pro jejich editaci, přejmenovávat je a skupiny odstraňovat (viz obr. 5.60). Skupin sloupců jsou dva druhy, obyčejná skupina a skupina se seskupením.

Obyčejná skupina znamená, že vyberu podmnožinu datových pohledů a ta se mi zobrazí v jedné skupině sloupců v datagridu.

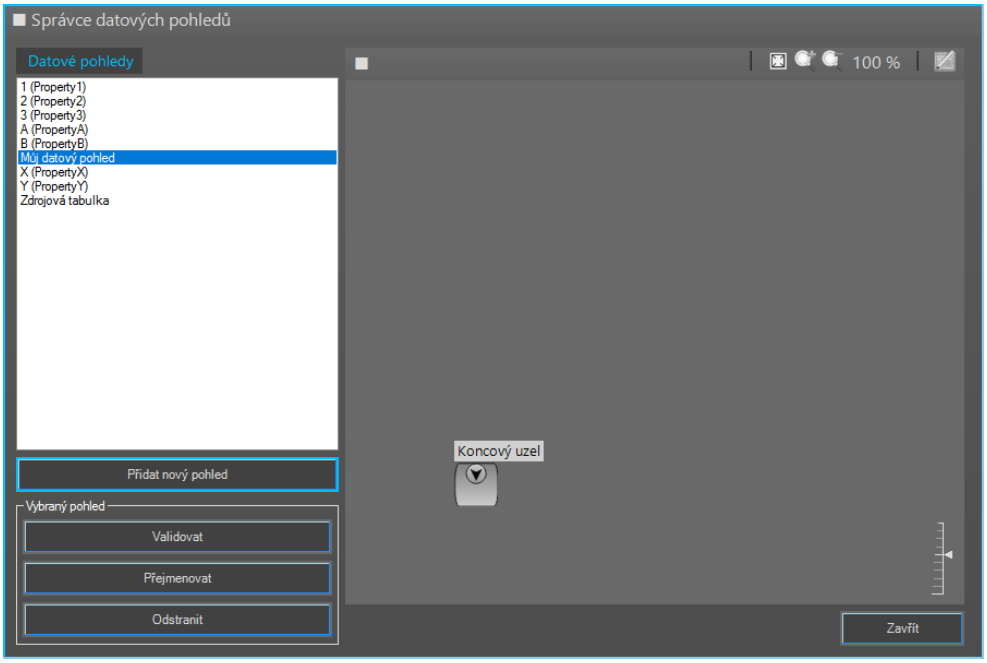
Skupina se seskupením znamená, že vyberu podmnožinu datových pohledů a dále také parametr, dle kterého se mají skupiny utvořit (viz obr. 5.61) a výsledkem je pak několik skupin sloupců. Tyto skupiny jsou utvořeny tak, že v každé z nich jsou sloupce / časové řady se stejnou hodnotou parametru, dle kterého se měly utvořit skupiny. Tato skupina byla použita v ukázce druhého příkladu, kde byly ve skupině vybrány sloupce: „Alza – počet návštěv“, „Alza – počet objednávek“, „Mironet – počet návštěv“, „Mironet – počet objednávek“ atd. a parametr seskupení byl „Druh časové řady“, tedy zda se jedná o počet návštěv nebo objednávek. Výsledkem pak byly dvě skupiny sloupců, první skupina se sloupci s návštěvami a druhá skupina se sloupci s objednávkami.



Obrázek 5.60: Správa skupin sloupců datagridu



Obrázek 5.61: Nastavení skupiny sloupců se seskupením



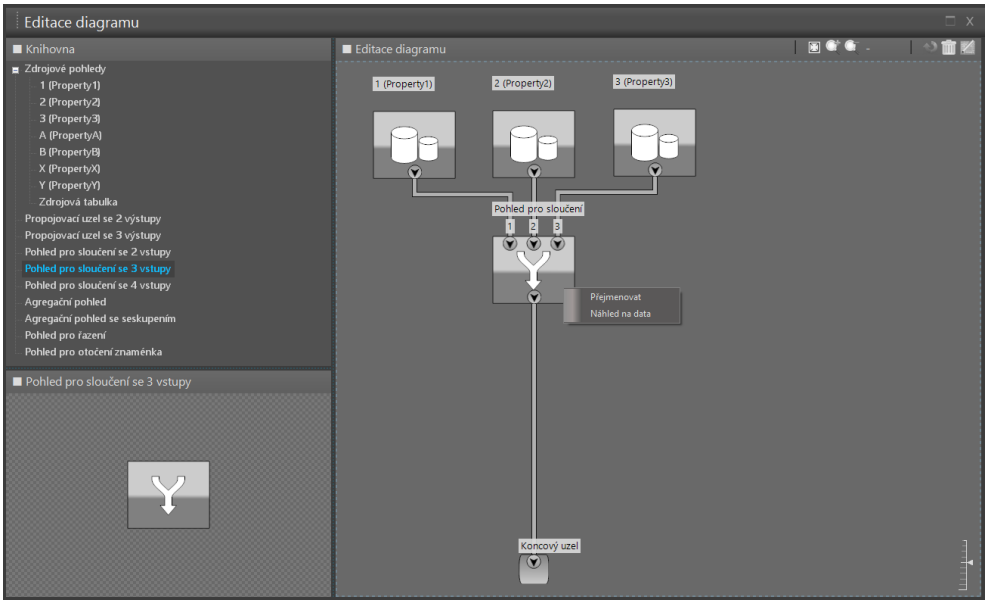
Obrázek 5.62: Prázdný diagram

5.2.5 Editace datových pohledů

Do správy datových pohledů (viz obr. 5.62) se lze dostat z editace vybraného schématu. Ve správě lze prohlížet dostupné datové pohledy a pro ty, které nejsou fixní, tedy pohled „Zdrojová tabulka“ a pohledy jednotlivých vlastností datové třídy, je možné zobrazit diagram, zobrazující strom datových pohledů. K editaci tohoto stromu pohledů lze přejít tlačítkem s ikonkou tužky vpravo nahoře. Dále zde lze tyto pohledy / stromy přidávat, přejmenovávat, odstraňovat a validovat (validace bude diskutována později).

Diagram je zobrazovací komponenta v naší aplikaci, která umožňuje práci s bločky, jejich konektory a jejich propojením – umožňuje tedy tvořit grafy (uzly a hrany), což je vhodné pro modelování stromů datových pohledů. V současné době se diagram v aplikaci používá i pro další účely (jako modelování algoritmů), takže se jedná o nástroj, na který je uživatel naší aplikace zvyklý.

Při editaci rozhodovacího stromu (viz obr. 5.63) je v levé horní části výběr dostupných datových pohledů. V levé dolní části je pak náhled na právě vybraný datový pohled, který lze umístit do diagramu přetažením myši. V hlavní části editace je samotný rozhodovací strom. Ve stromu jsou tedy již zmíněné bločky datových pohledů. Datové pohledy zde mají vstupy a výstupy, které je možné propojovat a tvořit tak vztahy mezi pohledy odpovídající stromové struktuře. Pohledy dále mají kontextové menu, které lze pravým tlačítkem myši na tyto pohledy otevřít a pomocí kterého lze jednotlivé pohledy nasta-



Obrázek 5.63: Jednoduchý strom v diagramu

vovat.

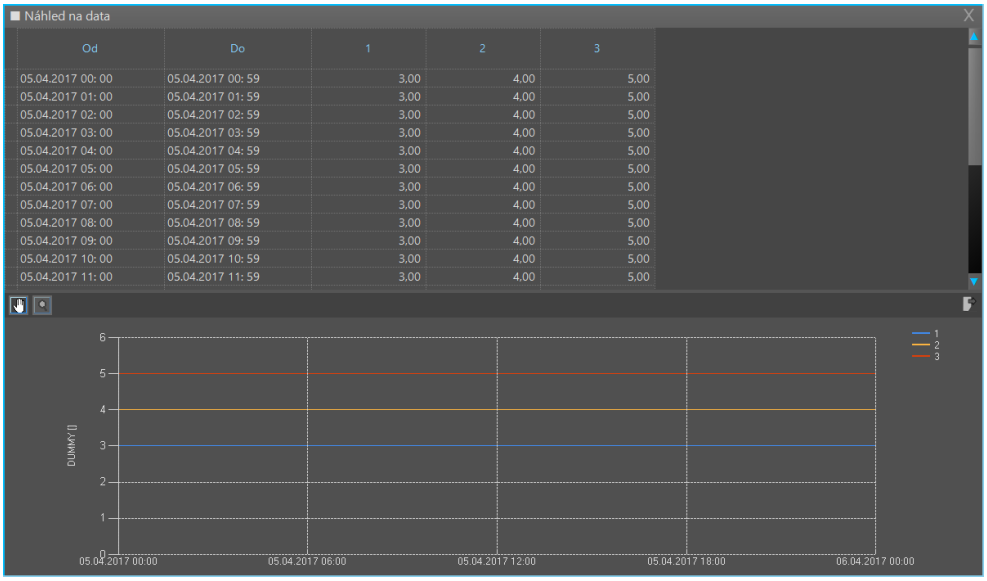
Strom datových pohledů má jediný výstup a to pohled, který je propojen s koncovým uzlem – na obrázku je to pohled pro sloučení. Strom datových pohledů dále musí splňovat podmínky, které jsou diskutovány dále v podkapitole o validaci 5.2.6.

Kromě nastavení lze k datovému pohledu přes jeho kontextové menu také zobrazit konkrétní data (viz obr. 5.64) datového objektu (toho, který je v aplikace právě vybrán). Náhled na data funguje tak, že se zpracuje celý diagram pohledů, stejně jako kdyby se schéma uložilo (vygeneruje se tedy XML Předpis atd.), pouze namísto okolního schématu je vytvořeno speciální dočasné schéma, které obsahuje jednu záložku v grafu a v datagridu a zobrazuje právě ten pohled, na který měl být náhled na data zobrazen. Toto schéma se neukládá do souborů na disk, jako tomu je při uložení běžného schématu, ale rovnou pošle aplikaci ke zpracování a k zobrazení.

5.2.6 Validace stromu datových pohledů

Strom datových pohledů (graf) musí splňovat následující podmínky, které jsou při validaci kontrolovány:

1. Graf obsahuje alespoň dva uzly (krom koncového) – graf s jedním uzlem by znamenal propojení zdrojového pohledu s koncovým uzlem, což je identické s přímým použitím zdrojového uzlu. Takový diagram tedy nemá smysl vytvářet.



Obrázek 5.64: Náhled na data datového pohledu

2. Strom je acyklický (cykly myšleno orientované). Strom datových pohledu s cykly nedává smysl a při zpracování by došlo k chybě.
3. Všechny konektory (vstupy a výstupy) jsou zapojeny.
4. Celý graf je slabě souvislý – tedy neobsahuje žádný osamocený podgraf nepropojený se zbytkem grafu. Tento podgraf by v diagramu neměl žádný význam.

Kontrola acykličnosti (podmínka 2) je implementována algoritmem postupného odebírání kořenů (uzlů bez vstupů) grafu [27]. Zjednodušený popis algoritmu je následující:

1. V grafu najdu libovolný uzel, bez vstupů (kořen) – každý acyklický graf musí alespoň jeden obsahovat.
2. Tento uzel a všechny z něj vedoucí hrany z grafu odeberu.
3. V grafu, který mi zůstal, opakuji stejný postup.
4. Pokud je graf acyklický, podaří se mi tímto způsobem z grafu odstranit všechny uzly.

Při bližší analýze podmínek se dále ukázalo, že podmínka 4 vyplývá z podmínek 2 a 3. Není ji tedy potřeba kontrolovat (pouze je v kódu ověřeno, že platí pomocí aserce). Na závěr této podkapitoly následuje náznak důkazu tohoto tvrzení. Dokazuji tedy, že plně zapojený, acyklický graf pohledů je vždy slabě souvislý:

1. Krom koncového uzlu mají všechny uzly alespoň 1 výstup.
2. Každé acyklické (podmínka 2), slabě souvislé zapojení nekoncových uzlů (dále Zapojení) má tedy alespoň jeden nezapojený výstup.
3. Koncový uzel má jeden vstup a žádný výstup a v grafu je vždy právě jeden.
4. Všechny vstupy i výstupy musí být zapojeny (podmínka 3), tím pádem i tento výstup musí být zapojen a jediný způsob, jak ho zapojit, aniž by vznikl další nezapojený výstup, je do koncového uzlu.
5. Kdyby existovalo nějaké další Zapojení, vždy by mělo alespoň jeden nezapojený výstup, což je v rozporu s předpokladem, že jsou všechny zapojeny.
6. Kdyby naopak neexistovalo žádné Zapojení, v grafu stále zbývá nezapojený koncový uzel.
7. Zapojení tedy musí být právě jedno, propojeno s koncovým uzlem a celý graf je tedy souvislý.

5.2.7 Generování Předpisu ze stromu datových pohledů

Při tvorbě Předpisu, resp. tvorbě jednotlivých stromů v XML je potřeba vyřešit problém, že stromy v XML jsou stromy striktně vzato, zatímco jak již bylo několikrát diskutováno, stromy datových pohledů je nepřesné označení a ve skutečnosti se jedná o acyklické grafy. Tyto acyklické grafy je tedy potřeba nějakým způsobem mapovat na skutečné stromy.

Sestrojil jsem algoritmus, který umožňuje jeden acyklický graf mapovat na několik stromů, resp. acyklický graf na tyto stromy dekomponuje. Algoritmus předpokládá, že graf je validní (viz validace v předchozí podkapitole). Ve zbytku podkapitoly následuje popis tohoto algoritmu.

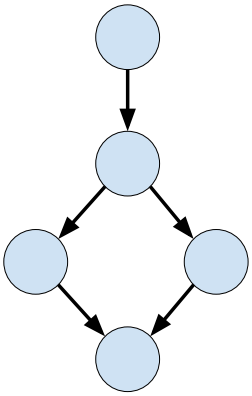
Aby nebyl popis algoritmu příliš abstraktní, popíši jej na příkladu.

Znázorněný graf (viz obr. 5.65) je strom datových pohledů (zjednodušený, že v něm nejsou vykresleny propojovací a koncové uzly). Pohledy bez vstupů jsou zdrojové pohledy, pohledy s více vstupy jsou pohledy pro sloučení a pro zjednodušení předpokládejme, že ostatní pohledy (s právě jedním vstupem) jsou agregační.

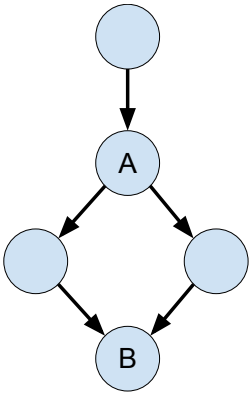
Začnu tím, že v grafu označím kořeny budoucích XML stromů. Kořenem bude každý uzel, který je buď koncový, nebo má více než jednoho potomka (viz obr. 5.66).

Při tvorbě XML stromů pak procházím kořeny v libovolném pořadí a píši pro ně XML strom (náznakem XML syntaxe, znak „]“ značí uzavření všech otevřených elementů).

Pro strom A:



Obrázek 5.65: Dekompozice prvního stromu

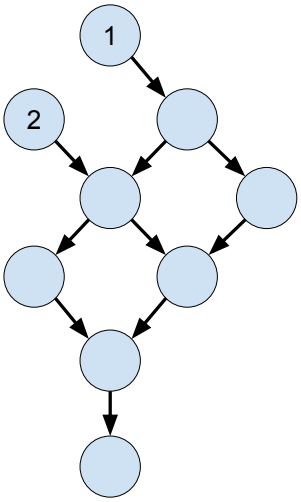


Obrázek 5.66: Dekompozice prvního stromu s označenými kořeny

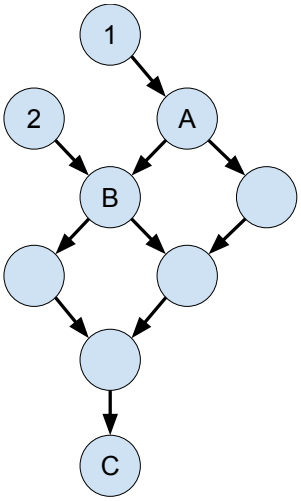
```
<strom A>
  <agregace>
    <zdroj dat />
]
```

Pro strom B, pak jakmile narazím na kořen stromu A, umístím na strom A pouze odkaz:

```
<strom B>
  <sloučení>
    <agregace>
      <strom A~/>
    </agregace>
      <agregace>
        <strom A~/>
]
```



Obrázek 5.67: Dekompozice druhého stromu



Obrázek 5.68: Dekompozice druhého stromu s označenými kořeny

Při psaní XML tedy postupuji od kořene proti směru hran k listům. Navštívím-li nějaký kořen, umístím místo něj odkaz na příslušný strom (nevadí, pokud zatím „neexistuje“) nebo navštívím-li zdroj dat, také na něj umístím odkaz. Z toho vyplývá, že zdroje dat a kořeny jiných stromů jsou listy.

Algoritmus předvedu ještě jednou na složitějším příkladě.

Tento příklad (viz obr. 5.67) má dva zdroje dat, aby je bylo možné odlišit, jsou očíslovány. Začnu tedy označením kořenů stromů (uzly s dvěma a více potomky nebo koncový uzel – viz obr. 5.68).

A dále již jen procházím kořeny a sestavuji XML stromy:

```
<strom A>
  <agregace>
    <zdroj dat 1 />
]

<strom B>
  <sloučení>
    <zdroj dat 2 />
    <strom A~/>
]

<strom C>
  <agregace>
    <sloučení>
      <agregace>
        <strom B />
      </agregace>
      <sloučení>
         <strom B />
         <agregace>
           <strom A~/>
]
```

Dekompozice druhého příkladu je tedy hotova.

5.2.8 Přehled implementovaných tříd

V této podkapitole bude probrána implementace nástroje na úrovni implementovaných tříd v programovacím jazyce.

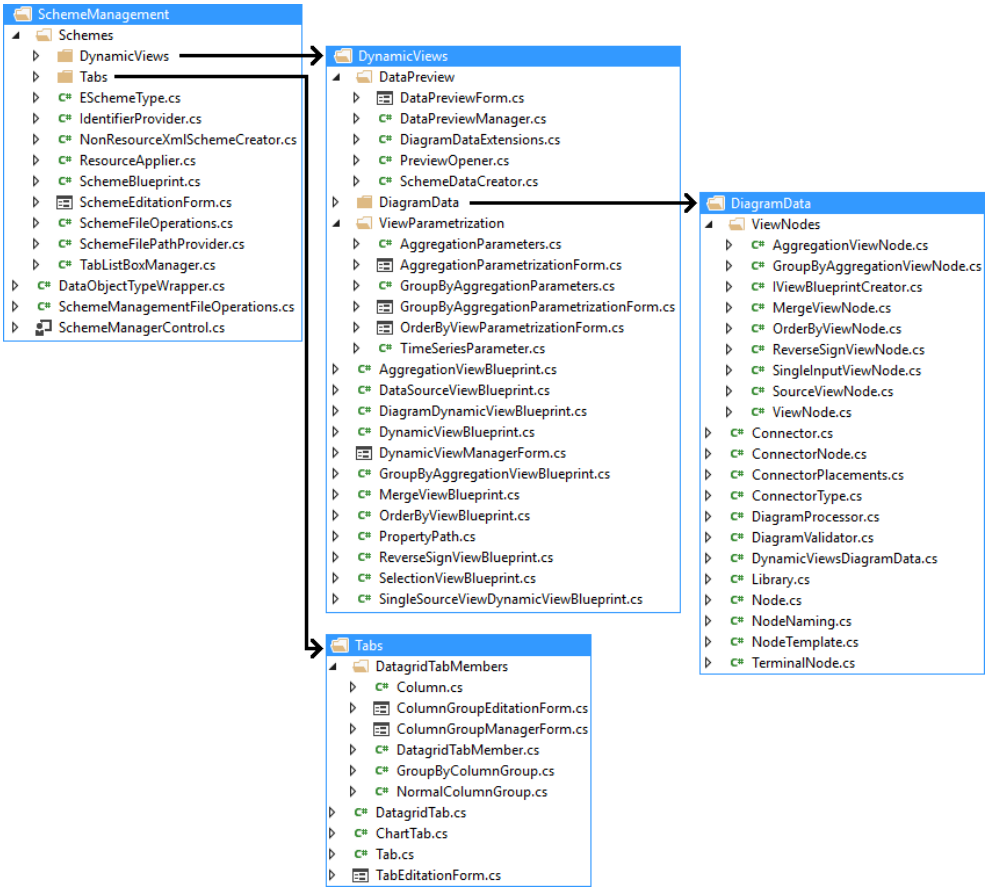
Nástroj byl implementován v celkem 66 třídách (viz obr. 5.69). Popis jednotlivých tříd včetně diagramů, které znázorňují metody a atributy tříd a také vztahy mezi třídami lze nalézt v dokumentaci na přiloženém DVD.

Složky na obrázku reprezentují rozdělení tříd do jmenných prostorů (namespaces). Tabulka 5.1 přiřazuje jednotlivým těmto jmenným prostorům doménu, kterou v nástroji řeší.

Pro představu o rozsahu implementace uvedu několik statistik ohledně počtu řádků těchto tříd. Počty řádek jednotlivých tříd jsou uvedeny v soubory na přiloženém DVD, tabulka 5.2 uvádí pouze shrnující hodnoty.

5.2.9 Code Contracts

Při implementaci byly použity metody pro ověření správnosti implementace, mezi které patří aserce (asserts), preconditions a postconditions jednotlivých metod a invarianty objektů (object invariants) jednotlivých tříd. Jedná se



Obrázek 5.69: Rozdělení tříd do jmenných prostorů

Tabulka 5.1: Popis jmenných prostorů

Jmenný prostor	Doména
SchemeManagement	Správa schémat
Schemes	Editace vybraného schématu
Tabs	Editace záložek datagridu a grafu vybraného schématu
DatagridTabMembers	Editace skupin sloupců v záložce datagridu
DynamicViews	Správa a editace datových pohledů
DataPreview	Náhled na data při editaci datových pohledů
ViewParametrization	Parametry datových pohledů a formuláře pro jejich editaci
DiagramData	Editace datových pohledů v diagramu
ViewNodes	Uzly jednotlivých datových pohledů v diagramu

Tabulka 5.2: Statistiky kódu

Počet tříd	66
Celkový počet řádků	4924
Celkový počet řádků	74,6
Počet řádků nejdelší třídy	214

o předpoklady o stavu programu, které jsou ověřeny za běhu a v případě jejich neplatnosti, program zobrazí chybu, aby se programátor o této nesrovnalosti dozvěděl co nejdříve a nejblíže její příčině. Implementaci těchto metod pro ověření správnosti zajišťuje nástroj Microsoft Code Contracts [28].

Aserce jsou umístěny v těle metod a testují tak správnost na daném řádku vykonávané metody.

Následuje ukázka použití aserce při ověření podmínky ve validaci stromu datových pohledů, kontrolující, že acyklický (metoda DoesDiagramContainCycle), plně zapojený (metoda AreAllConnectorsConnected) strom datových pohledů je slabě souvislý (metoda AreAllNodesReachable):

```
if (DoesDiagramContainCycle())
{
    result.ReportError(ObjectLocalization.DiagramMustBeAcyclic);
}
else
{
    Contract.Assert(
        !AreAllConnectorsConnected() || AreAllNodesReachable());
}
```

Preconditions a postconditions jsou umístěny v hlavičce metody a testují správnost programu před vykonáním dané metody (preconditions) nebo po návratu z dané metody (postconditions). Pomocí postconditions lze zkontrolovat i návratovou hodnotu metody.

Následuje ukázka použití preconditions a postconditions. Precondition je použita pro ověření, že existuje cesta k zadanému schématu před jeho načtením a postcondition je použita pro ověření, že nově zkonstruované schéma je validní.

```
internal static SchemeBlueprint LoadSchemeBlueprint(string path)
{
    Contract.Requires(DataProvider.FileExists(path));
    ...

internal SchemeBlueprint(Type dataObjectType, string name)
{
```

```
    Contract.Ensures(Validate().IsValid);
    ...
```

Invarianty objektu jsou umístěny v pro ně vyhrazené metodě dané třídy a jejich kontrola probíhá před a po každém volaní libovolné veřejné metody této třídy. Typicky se v nich pak kontroluje stav objekty, který je vždy platný (např., že daná vlastnost nabývá pouze povolených hodnot).

Následuje ukázka invariantu třídy reprezentující uzel datového pohledu v diagramu, který předpokládá, že v každém tomto uzlu je vždy právě jeden výstup.

```
[ContractInvariantMethod]
private void ObjectInvariant()
{
    Contract.Invariant(
        GetConnectors(ConnectorType.Output).Count() == 1);
}
```

Kapitola 6

Dokumentace

Tato kapitola se zabývá dokumentací implementace vytvořeného nástroje.

Při dokumentaci byl dodržován princip upřednostnit sebepopisný, čitelný kód před kódem bez těchto kvalit, který je nutné doprovázet komentáři [29]. Tento princip spočívá v tom, že je lepší zvolit k popisu implementace prostředky dostupné v programovacím jazyce, jako vhodné strukturování na rozumně malé a jednoduché celky (metody a třídy) a zvolit jim vhodné, vypovídající pojmenování, než jen příliš složitý a špatně čitelný kód doprovázet komentáři. Komentáře totiž často kód duplikují a při dalším vývoji se stávají neaktuálními a tím pádem i potenciálně nepravdivými. Nejspolehlivější popis toho, co program dělá je tak samotný kód.

Komentáře v kódu jsou tedy zvoleny jen zřídka, v případě nenalezení vhodnějšího způsobu dokumentace pomocí prostředků kódu. Na základě tohoto principu je tak dokumentace především na úrovni kódu, který je členěn do mnoha tříd a metod, jejichž názvy by měly vystihovat jejich význam.

Kromě tohoto přístupu je však ke každé třídě napsán krátký XML komentář, popisující třídu podrobněji a dále popisující její kontext a vztahy k ostatním třídám.

Dokumentace byla z kódu a jeho komentářů vygenerována pomocí nástrojů Doxygen a Graphviz a je umístěna na přiloženém DVD.

Jako dokumentací některých souvislostí v implementaci lze také považovat použití asercí, preconditions, postconditions a invariantů (viz podkapitola 5.2.9). Jejich použití totiž obsahuje dodatečné informace o stavu programu při vyhodnocení daných částí kódu.

Kapitola 7

Použité nástroje

Při rešerši byly zkoumány programy SSRS Report Builder [30], Dashboard Syncfusion [31] a Sisense [32]. Při jejich průzkumu byla v Microsoft SQL Server [33] připravena testovací SQL databáze. Jako vývojové prostředí bylo použito Microsoft Visual Studio [34], které napomohlo s organizací souboru programu, asistencí při psaní kódu, ladícími nástroji a nástroji pro kompilaci programu. K vygenerování dokumentace z implementace byly použity nástroje Doxygen [35] a Graphviz [36]. Pro sazbu textu diplomové práce byl použit systém LaTeX.

Závěr

Cílem práce bylo nastudovat možné způsoby vizualizace jazyků pro zpracování dat podobných internímu jazyku ve stávající aplikaci, a dále pak implementace GUI pro vizualizaci tohoto jazyka. Toto GUI mělo umožnit definovat datové pohledy pro výběr, třídění a agregace časových řad, umožnit tyto datové pohledy řetězit a skládat do stromových struktur a tyto stromové struktury mapovat do tabulek a grafů pro výsledné zobrazení uživateli. Vzhledem k tomu, že je GUI součástí aplikace, na které pracují i další vývojáři, byl také požadavek na kvalitu a pečlivost implementace. Výstupem dále měla být i vhodná dokumentace veřejných rozhraní implementovaných tříd.

V práci se podařilo realizovat všechny vytyčené cíle.

V první části práce proběhl průzkum několika podobných nástrojů, zabývajících se předpisem pro zpracování dat a dále bylo implementováno GUI, které je součástí aplikace. Při průzkumu byly nalezeny některé pěkné vlastnosti zkoumaných nástrojů, které byly realizovány v implementovaném GUI. Mezi tyto vlastnosti patří např.:

- zobrazení náhledu na konkrétní data již při tvorbě obecného schématu,
- přehledná tvorba předpisu názvu pro agregační sloupce
- nebo definice datových pohledů předcházející a zároveň nezávislou na zvolené zobrazovací komponentě.

Implementované GUI umožňuje z datových struktur aplikace vybírat časové řady a vytvářet datové pohledy pro agregace, seskupování, řazení a další operace s časovými řadami. Tyto pohledy lze vzájemně propojovat a utvářet tak stromy datových pohledů. Tyto stromy lze dále mapovat na zobrazovací komponenty aplikace, tedy grafy a tabulky. V nástroji lze definovat i další parametry těmto zobrazovacím komponentám, jako popisky os v grafech nebo skupiny sloupců v tabulkách.

GUI je uživatelsky přívětivé a obsahuje validace uživatelských vstupů, aby se zamezilo chybám a neočekávanému chování. Za použití sestrojených algoritmů se z objektové struktury schématu (obsahující v GUI vytvořené stromy datových pohledů apod.) stává poměrně jednoduchý a čitelný zápis v interním doménově specifickém jazyce (v porovnání s podobnými zápisy ostatních nástrojů v rešeršní části).

Implementace proběhla pečlivě, s důrazem na čitelnost kódu, omezení jeho duplikace a vhodné členění do jednotlivých tříd a metod. Tyto metody, třídy a další prvky programovacího jazyka byly vhodně pojmenovány, aby dobře dokumentovaly svůj význam a dále byly v implementaci použity techniky pro ověření správnosti programu Code Contracts. Implementace je poměrně rozsáhlá, čítající 66 tříd o téměř 5000 řádcích kódu. Součástí práce je i dokumentace veřejných rozhraní implementovaných tříd.

Implementace byla úspěšně dokončena, nástroj je v současné době používán ostatními vývojáři aplikace a plánuje se jeho další vylepšování a zpřístupnění běžnému uživateli (čemuž předchází detailnější otestování a odladění méně častých případů použití).

Literatura

[1] Klíma, J.: Implementace modulu pro zpracování datových struktur. 2015.

[2] Corporation, M.: What is SQL Server Reporting Services (SSRS)? [online], [cit. 21. 4. 2017]. Dostupné z: https://msdn.microsoft.com/en-us/library/ms159106.aspx

[3] Corporation, M.: Reporting Services Concepts (SSRS). [online], [cit. 21. 4. 2017]. Dostupné z: https://msdn.microsoft.com/en-us/library/bb630404.aspx

[4] Corporation, M.: Designing Reports in Report Designer and Report Builder 3.0 (SSRS). [online], [cit. 21. 4. 2017]. Dostupné z: https://technet.microsoft.com/cs-cz/library/ms159253(v=sql.105).aspx

[5] Corporation, M.: Introducing Business Intelligence Development Studio. [online], [cit. 21. 4. 2017]. Dostupné z: https://msdn.microsoft.com/en-us/library/ms173767(v=sql.105).aspx

[6] Corporation, M.: Report Builder in SQL Server 2016. [online], [cit. 21. 4. 2017]. Dostupné z: https://msdn.microsoft.com/en-us/library/dd220460.aspx

[7] Corporation, M.: Tutorial: Create a Quick Chart Report Offline (Report Builder). [online], [cit. 21. 4. 2017]. Dostupné z: https://msdn.microsoft.com/en-us/library/dd220476.aspx

[8] Corporation, M.: Prerequisites for Tutorials (Report Builder). [online], [cit. 21. 4. 2017]. Dostupné z: https://msdn.microsoft.com/en-us/library/ff519556.aspx

[9] Corporation, M.: SQL Server RDL Specification. [online], [cit. 21. 4. 2017]. Dostupné z: https://msdn.microsoft.com/library/dd297486(SQL.100).aspx

[10] Corporation, M.: XML Query Syntax for XML Report Data (SSRS). [online], [cit. 21. 4. 2017]. Dostupné z: https://msdn.microsoft.com/en-us/library/ms345251.aspx

[11] Corporation, M.: Reporting Services Tutorials (SSRS). [online], [cit. 21. 4. 2017]. Dostupné z: https://msdn.microsoft.com/en-us/library/bb522859.aspx

[12] Corporation, M.: Expression Uses in Reports (Report Builder and SSRS). [online], [cit. 21. 4. 2017]. Dostupné z: https://msdn.microsoft.com/en-us/library/ms345237.aspx

[13] Corporation, M.: Expression Examples (Report Builder and SSRS). [online], [cit. 21. 4. 2017]. Dostupné z: https://msdn.microsoft.com/en-us/library/ms157328.aspx

[14] Corporation, M.: Query Design Tools (SSRS). [online], [cit. 21. 4. 2017]. Dostupné z: https://msdn.microsoft.com/en-us/library/ms345246.aspx

[15] Corporation, M.: Relational Query Designer User Interface (Report Builder). [online], [cit. 21. 4. 2017]. Dostupné z: https://msdn.microsoft.com/en-us/library/dd220607.aspx

[16] Corporation, M.: Date and Time Data Types and Functions (Transact-SQL). [online], [cit. 21. 4. 2017]. Dostupné z: https://msdn.microsoft.com/en-us/library/ms186724.aspx

[17] Corporation, M.: float and real (Transact-SQL). [online], [cit. 21. 4. 2017]. Dostupné z: https://msdn.microsoft.com/en-us/library/ms173773.aspx

[18] w3resource: SQL Arithmetic Operators. [online], [cit. 21. 4. 2017]. Dostupné z: http://www.w3resource.com/sql/arithmetic-operators/sql-arithmetic-operators.php

[19] Technologies, S.: Comparing Crystal Reports and SQL Server Reporting Services. [online], [cit. 21. 4. 2017]. Dostupné z: http://www.seguetech.com/comparing-crystal-reports-sql-server/

[20] Dundas Data Visualization, I.: Dashboards vs. Reports: Which One Should You Go With? [online], [cit. 21. 4. 2017]. Dostupné z: http://www.dundas.com/support/blog/dashboards-vs.-reports-which-one-should-you-go-with

[21] Syncfusion, I.: Syncfusion Dashboard Platform – Dashboard designer walk-through. [online], [cit. 21. 4. 2017]. Dostupné z: https://www.youtube.com/watch?v=caMB3K3CsHo

[22] Syncfusion, I.: The Syncfusion Dashboard Platform: Building and Deploying Dashboards. [online], [cit. 21. 4. 2017]. Dostupné z: https://www.youtube.com/watch?v=vUHdwvZIQuY

[23] Sisense, I.: Create a New ElastiCube Schema. [online], [cit. 21. 4. 2017]. Dostupné z: https://www.youtube.com/watch?v=ultJDuWPdCI&list=PLQHiBfQxt7JO__Tyev8YFmtvp08pclDks

[24] Sisense, I.: Building ElastiCubes. [online], [cit. 21. 4. 2017]. Dostupné z: https://www.sisense.com/documentation/v5/elasticube-manager/importing-data-into-a-data-model/

[25] Sisense, I.: Data Files and Data Flow in Sisense. [online], [cit. 21. 4. 2017]. Dostupné z: https://support.sisense.com/hc/en-us/articles/230644108-Data-Files-and-Data-Flow-in-Sisense

[26] Sisense, I.: Error and Troubleshooting. [online], [cit. 21. 4. 2017]. Dostupné z: https://www.sisense.com/documentation/error-and-troubleshooting/

[27] Doc. RNDr. Josef Kolář, C.: Orientované grafy, reprezentace grafů. [online], [cit. 21. 4. 2017]. Dostupné z: https://edux.fit.cvut.cz/courses/BI-GRA/_media/lectures/gra-predn-03cb.pdf

[28] Corporation, M.: Code Contracts. [online], [cit. 21. 4. 2017]. Dostupné z: https://msdn.microsoft.com/en-us/library/dd264808(v=vs.110).aspx

[29] Martin, R. C.: *Clean Code: A Handbook of Agile Software Craftsmanship*. Upper Saddle River, NJ, USA: Prentice Hall PTR, první vydání, 2008, ISBN 0132350882, 9780132350884.

[30] Corporation, M.: Microsoft SQL Server Report Builder 2016. Version 2.1.0.2.

[31] Syncfusion: Syncfusion Dashboard. Version 2.1.0.2.

[32] Sisense: Sisense: Business Intelligence (BI) Software.

[33] Corporation, M.: Microsoft SQL Server 2016.

[34] Corporation, M.: Microsoft Visual Studio Community 2015 for Windows Desktop.

[35] van Heesch, D.: Doxygen. Version 1.8.9.1.

[36] Gansner, E. R.; North, S. C.: An open graph visualization system and its applications to software engineering. *SOFTWARE - PRACTICE AND EXPERIENCE*, ročník 30, č. 11, 2000: s. 1203–1233.

Příloha **A**

Seznam použitých zkratek

DSL Domain-specific language

GUI Graphical user interface

RDL Report Definition Language

SQL Structured Query Language

SSRS SQL Server Reporting Services

XML Extensible markup language

Příloha **B**

Obsah přiloženého DVD

- `Dalsi_materialy` ..... doplňující materiály k práci
  - `Nastroj_1_SSRS` ..... materiály k nástroji SSRS
  - `Nastroj_3_Syncfusion` ... materiály k nástroji Syncfusion Dashboard
  - `Nastroj_4_Sisense` ..... materiály k nástroji Sisense
  - `Testovaci_data` ..... data k testovacím příkladům
  - `Materialy_k_implementaci` ..... materiály ke kapitole implementace
- `DP_Klima_Jakub_2017.pdf` ..... práce ve formátu PDF
- `readme.txt` ..... stručný popis obsahu DVD
- `Zdrojova_forma_prace` ..... zdrojová forma práce ve formátu LaTeX
  - `Obrazky` ..... obrázky použité v práci